NAVIGATING THE LANGUAGE SEAS

March/April 1991 Volume 9.2

Programmer's Journal™

4.95 Canada $5.95

Paul Yao on Windows 3.0 C Compilers
page 28

Fortran 90 Is Coming
page 42

Exploring XMS in Pascal
page 72

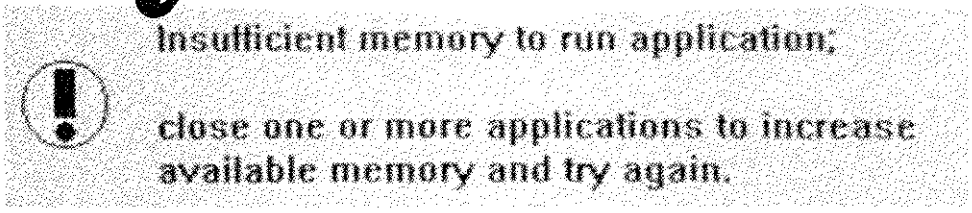
ASM: Using EMS for Fast File I/O
page 75

Porting to the Mac
page 60

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

386 and 486 Windows users:

DOS Lives. Despite what you may have read.



Windows 3.0 may have been a big step forward for some programs.

But it was a big step backward for DOS. Suddenly, it was 1987 all over again. Not enough room for DOS programs to run because TSRs, utilities, drivers and buffers were taking up room your DOS programs need.

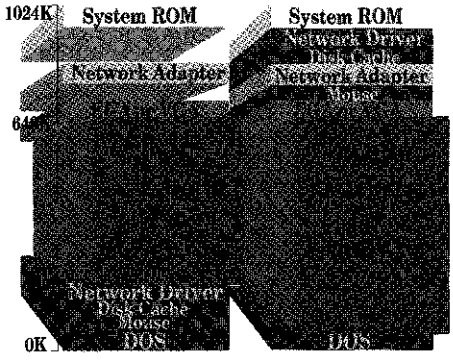
When Windows 3.0 says there's not enough room to run your DOS programs, it's just trying to tell you it needs QEMM 5.1.

QEMM 5.1 to the rescue

Now, we've updated QEMM to provide additional memory for DOS programs within Windows 3.0.

QEMM 5.1 works with the built-in capabilities of your 80386 or i486 processor to find and recover unused memory segments. As you can see in the chart, there are gaps in your PC's memory usage above 640K. QEMM 5.1 fills those gaps and provides more room for your DOS programs to run.

And of course, QEMM 5.1 still works with DOS when you're not running Windows. You get all the same benefits: up to 130K more memory to run the new generation of memory-hungry programs; space for larger spreadsheets and database files.



Before *After*

Unused memory is like hidden gold in your PC. QEMM finds it and makes it available.

Introducing DESQview 2.3 and DESQview 386 2.3 for users of Windows 3.0

They said it couldn't be done, but DESQview 2.3 can run Windows 3.0 programs. Not just in Windows "Real mode" but in "Standard" mode. That means programs can be up to 16MB.

And it can run DOS programs and DOS-extended programs i.e., 1-2-3 Release 3, side-by-side.

DESQview 386 2.3 does all that and more. It lets you run 386 DOS extended programs like AutoCAD 386 and IBM Interleaf side-by-side.

DOS, extended DOS; Windows—whatever standards you set, we will support. We're committed to helping you get the most out of your hardware and software today. And tomorrow.

QEMM-386 System Requirements: 80386-based PCs and PS/2s and PCs with 80386 add-in boards. Operating system: PC DOS 2.0-4.0, MS DOS 2.0-3.3, Windows 3.0. Conventional memory requirement 1.5K.
DESQview System Requirements: IBM Personal Computer and 100% compatibles (with 8086, 8088, 80286, or 80386 processors) with monochrome or color display; IBM Personal System/2 • Memory: 640K recommended; for DESQview itself (-145K • Expanded Memory (Optional): expanded memory boards compatible with the Intel AboveBoard; enhanced expanded memory boards compatible with the AST RAMpage; EMS 4.0 expanded memory boards • Disk: two diskette drives or one diskette drive and a hard disk • Graphics Card (Optional): Hercules, IBM Color/Graphics (CGA), IBM Enhanced Graphics (EGA), IBM PS/2 Advanced Graphics (VGA) • Mouse (Optional): Mouse Systems, Microsoft and compatibles • Modem for Auto-Dialer (Optional): Hayes or compatible • Operating System: PC-DOS 2.0-4.0; MS-DOS 2.0-3.3 • Software: Most PC-DOS and MS-DOS programs; programs specific to Microsoft Windows 1.03-2.1, GEM 1.1-3.0, IBM TopView 1.1 • Media: DESQview is available on either 5-1/4" or 3-1/2" floppy diskette.

Trademarks: Windows, MS-DOS: Microsoft Corporation; PS/2, Interleaf, TopView: IBM Corporation; 80386, i486, AboveBoard: Intel Corporation; 1-2-3: Lotus Development Corporation; AutoCAD 386: Autodesk, Inc; RAMpage: AST Research; Hercules; Mouse Systems; Hayes; GEM: Digital Research, Inc.

©1990 Quarterdeck Office Systems

Yes! I need increased DOS productivity in Windows!

Qty	Product	5-1/4	3-1/2	Each	Totals
	QEMM 386 5.1			$99.95	
	DESQview 386 2.3 (includes QEMM)			$219.95	
	DESQview 2.3			$129.95	
	Shipping & Handling $5 in USA/$10 outside USA				
	California Residents add 6.75%				
	Grand Total				

Payment method ☐ Visa ☐ MasterCard
Expiration ____/____
Card # _____
Name _____
Address _____
City _____
State ____ Zip _____

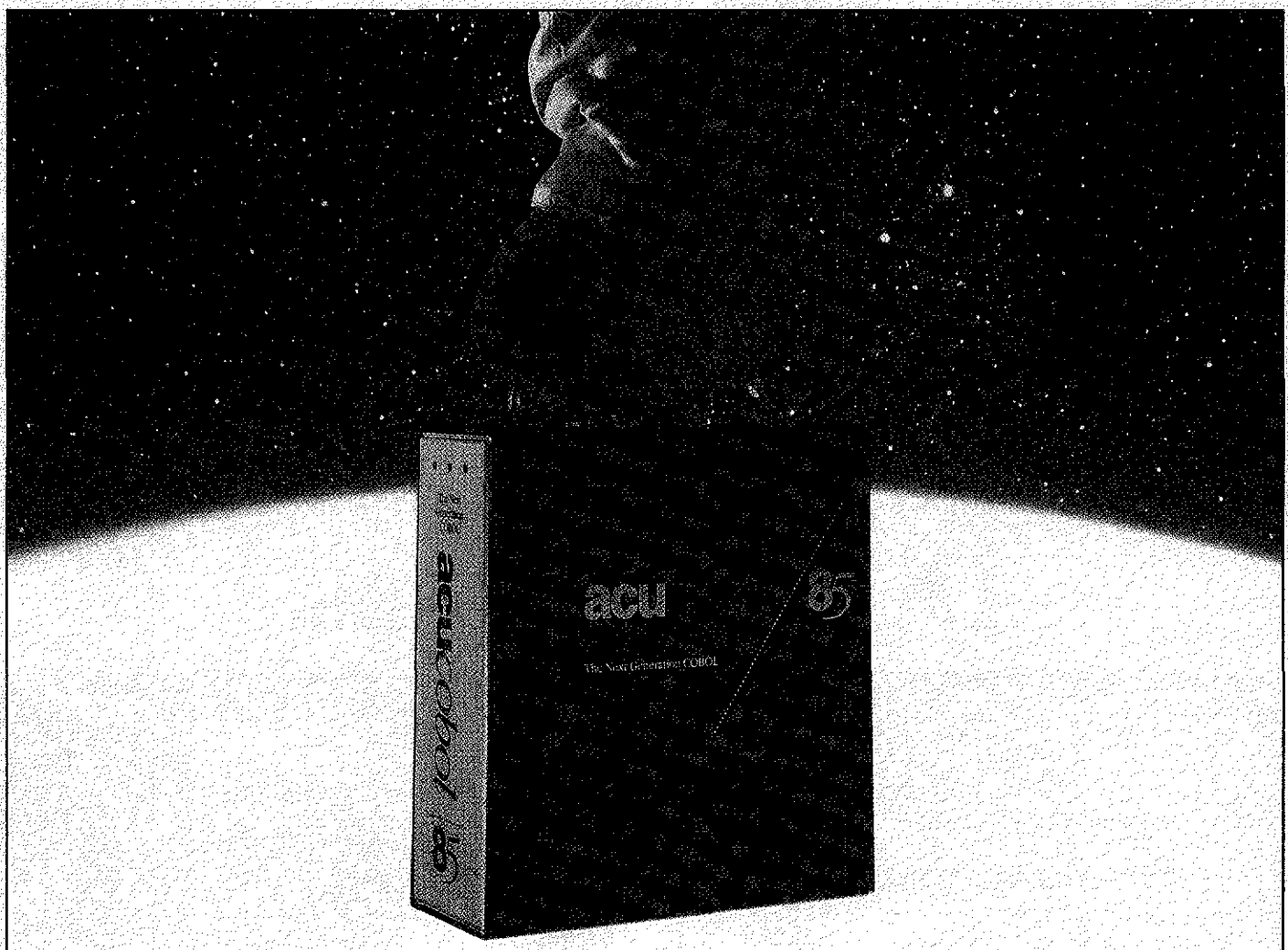
Please allow 3 weeks for delivery

Quarterdeck

Quarterdeck Office Systems, 150 Pico Blvd., Santa Monica, CA 90405 (213) 392-9851 Fax: (213) 399-3802

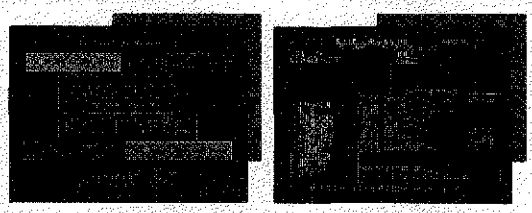
Circle 1 on Reader Service Card

PRODUCED BY BYTE.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Introducing The Most Portable COBOL In The Universe

Go Where You Want To Go



Transport COBOL programs to almost any platform—without changing a line of code. Run on AIX, MS DOS, NETBIOS, UNIX, AOS, OS/2, ULTRIX, XENIX, VMS—without recompiling. No other COBOL 85 offers such hardware independence or instantly-expanded possibilities. Plus, ACUCOBOL-85's completely compatible with RM/COBOL and VAX COBOL.

Open New Windows In Your Applications

Create better-looking programs. Add help screens, pop-up/pull-down menus, context-sensitive messages. ACUCOBOL-85 has a Display Window verb with extensions for boxed, reverse-video and pop-up windows in color. Paint special forms on the screen with our Display Line verb. And scroll up/down inside windows.

Don't Get Lost In Space

ACUCOBOL-85 is NBS certifiable error-free under High 1974 and Intermediate 1985 ANSI standards. And if you get off course, ACUCOBOL has an interactive source code debugger that's fast and easy to use.

Free Evaluation Copy

Call (619) 689-7220 today for your free evaluation copy of ACUCOBOL-85. And see the shape of things to come.

acucobol inc

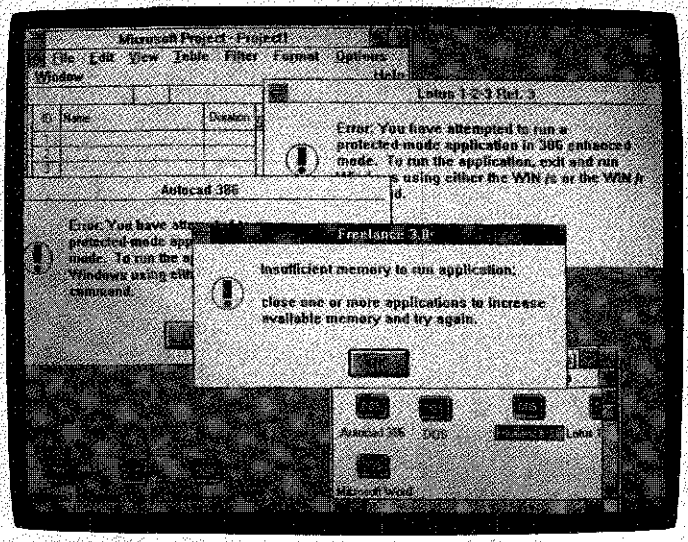
Voice: (619) 689-7220 Fax: (619) 566-3071
7950 Silverton Ave., Suite 201, San Diego, CA 92126

ACUCOBOL/ACUCOBOL-85 are trademarks of ACUCOBOL, Incorporated; MS-DOS and XENIX of Microsoft Corporation; NETBIOS/OS/2/AIX of International Business Machines Corporation; VMS/VAX/ULTRIX of Digital Equipment Corporation; UNIX of AT&T Bell Laboratories; RM/COBOL and RM/COBOL-85 of Ryan-McFarland Corporation.

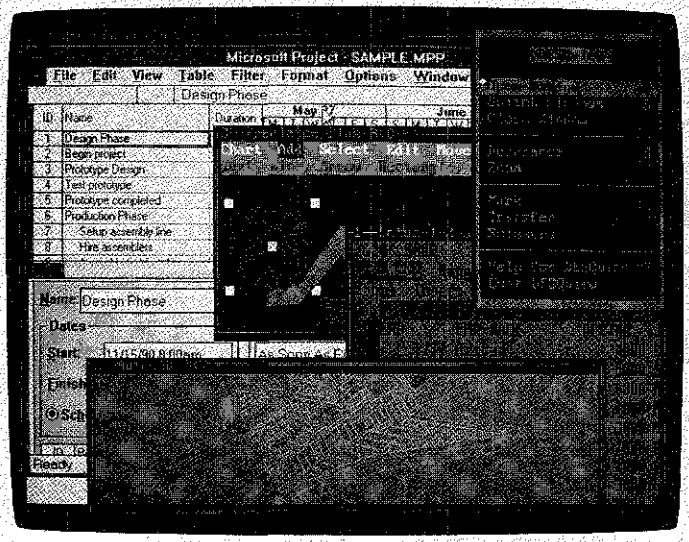
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Is Windows 3.0 the end of DESQview™?

Not if you're still using DOS programs.


Microsoft Project - Project1
File Edit View Table Filter Format Options Window Help
Lotus 1-2-3 Rel. 3
Autocad 386
Freelance 3.0
Insufficient memory to run application.
Close one or more applications to increase available memory and try again.
OK

Windows 3.0. The multitasking, windowing environment.



DESQview 386 2.3. The multitasking, windowing environment.

Are you still using MS-DOS programs on your PC?

You may want to use DESQview as your primary operating environment.

The new DESQview 2.3 and DESQview 386 2.3 let you use your favorite DOS and DOS-extended programs in windows side-by-side on 80286, 80386 and i486 PCs. As you can see above, you can even run Windows programs within DESQview 386. So the next time you get error messages like the ones at the left in Windows, remember how the same set of programs look running in DESQview.

Whatever standard you use—DOS, extended DOS or Windows—DESQview is still the best way to get the most out of the hardware and software you own today.

DESQview. The obvious choice.

Quarterdeck

Quarterdeck Office Systems, 150 Pico Boulevard, Santa Monica, CA 90405 (213) 392-9851 Fax: (213) 399-3802

This comparison was made using a system like the one you might run: Both shots show an ALR FlexCache 33/386 running DOS 3.3 with VGA display adaptor, Novell NetWare v3.01 Rev. A, with IPX/SPX v3.01 Rev A, Microsoft Mouse 7.00, and Microsoft SMARTDrive v3.03 disk cache. Buffers were set to 20. For the Windows screen, we ran Microsoft Windows 3 HIMEM.SYS and EMM386.SYS. For the DESQview screen, we ran QEMM 386 v5.1.

Trademarks: Microsoft, Windows, MS-DOS, 80386, i486, ALR, FlexCache, Novell, NetWare. ©1990 Quarterdeck Office Systems

Circle 3 on Reader Service Card.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Do You Hate COBOL?

Then the COBOL Report Activator *(COBRA)* System is for you!

The ***COBRA*** System is a keyword driven, free-form method to generate complete COBOL programs! Use the ***COBRA*** System and any COBOL compiler to write reports or reformat files. Just a few short, SQL-like statements and the program is written for you!

- ✓ No Divisions to worry about!
- ✓ No logic to figure out!
- ✓ No syntax/compile errors to fix!
- ✓ No logic errors to debug!
- ✓ No hassle!

The FD Conversion Utility (included) will convert existing COBOL File Descriptions into the ***COBRA*** System Data Dictionary Format automatically.

The ***COBRA*** System is easy to learn and use. With the examples in the manual and online HELP file, you can generate your first program in less than one hour from opening the package.

Do You Love COBOL?

Then the COBOL Report Activator *(COBRA)* System is for you!

The ***COBRA*** System generates textbook quality, structured, 'Quick and Clean' COBOL programs in seconds. Say goodbye to 'quick and dirty' programming forever!

With the ***COBRA*** System you can:

- ☆ Use your existing copybook members
- ☆ Use sequential, indexed and BTRIEVE files
- ☆ Use one or two files for input
- ☆ Add COBOL Code in seven places before you generate a COBOL program
- ☆ Modify the generated COBOL program
- ☆ Tailor the System defaults to fit your site

The ***COBRA*** System will generate code for internal sorts, audit trail reports, and up to five output files. Record selection logic is easy - include or exclude records as desired.

The **new** Version 4.0 of the ***COBRA*** System is available in both PC and mainframe versions.

Listed below are the statements required for an internally sorted, control break totals report, using a pre-defined file (from your COBOL FD):

```
TITLE SAMPLE REPORT FOR MULTIPLE CONTROL BREAK PROGRAM
REPORT TOTALS
SELECT EMPLOYEE-NBR, EMPLOYEE-LAST-NAME, EMPLOYEE-FIRSTNAME,
ADDRESS, THIS-PER-EARNINGS, YEAR-TO-DATE-EARNINGS
FROM EARNINGS-FILE
ORDER BY PLANT-CODE, DEPT-NBR, EMPLOYEE-NBR
GROUP BY PLANT-CODE, DEPT-NBR
```

☞ Special Half Price Offer for P. J. Readers!

Regular price for the ***COBRA*** System is $495. Pay just $247.50, with our No-risk, 30-day money back guarantee. Because of the special low price, we will accept only checks and money orders.

Olivadoti Associates, Inc.
(Formerly *COBRA* Systems Software)
P.O. Box 750817
Petaluma, CA 94975-0817 **(707) 763-5123**

Supports Realia, mbp, Microsoft/MicroFocus, R-M and Acucobol. Requires DOS 2.1 or higher, 320K, two disk drives or Hard disk.

Yes! I want 'Quick and Clean COBOL'®

Name _____
Street Address _____
City _____ State _____ Zip _____
Number of units ordered _____ @ $247.50 _____
Shipping ($5.00/unit) _____
Foreign Shipping (please add $4.00/unit) _____
California Residents add sales tax ($14.94/unit) _____
Total Amount enclosed: $_____

Limited time Offer - Expires June 1, 1991

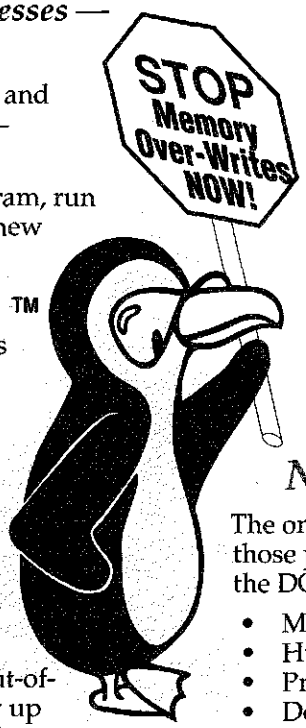
Contact this advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

POWER DEBUGGING

BOUNDS-CHECKER

Finds out-of-bounds memory accesses — AUTOMATICALLY.



Flush out those Nasty pointer problems and other out-of-bounds memory accesses — AUTOMATICALLY.

Each time you make a change to a program, run BOUNDS-CHECKER while testing the new code. If you accidentally access out-of-bounds memory, BOUNDS-CHECKER will pop up displaying the offending SOURCE LINE. And your program runs at full speed.

"BOUNDS-CHECKER and Soft-ICE make sophisticated use of the most powerful versions of Intel's processor family to track down some of DOS programming's most insidious bugs. If you're developing programs for DOS, these are essential tools."
PC Magazine
July, 1990 pg. 48

Ship Bug-Free Products

You can run BOUNDS-CHECKER while testing your program. There are no additional steps to your testing cycle, but you can feel secure when the program has passed through BOUNDS-CHECKER with no reported problems.

Many over-write problems and other out-of-bounds memory accesses do NOT show up during normal testing. An out-of-bounds memory location may be modified, but that particular location doesn't happen to be important at the time. Once the program is in the field and a certain network is loaded or a certain T&SR or device driver is loaded, that memory location suddenly becomes very important... AND THE SYSTEM CRASHES.

You can prevent these problems by making BOUNDS-CHECKER a standard part of your testing procedure.

Gives you the protection of a protected operating system under MS-DOS.

BOUNDS-CHECKER uses the 386 virtual machine technology to provide real-time memory protection. In addition BOUNDS-CHECKER uses the symbolic information output by your compiler to differentiate CODE and DATA. When your program is running, BOUNDS-CHECKER protects the program's CODE and all memory outside your program.

Requires 386 PC.
MS-DOS is a trademark of Microsoft Corporation.
386 is a registered trademark of Intel Corp.

Soft-ICE 2.5
New Version, New Features

The only debugger specifically designed to solve those problems unique to MS-DOS that we call the DOS Nasties.

- Memory over-writes
- Hung programs
- Program too big to debug
- Debugging T&SRs and Loadable Drivers
- Multiple Symbol Tables
- Supports Microsoft C 6.0 & Turbo C++

Windows Developers
Run CODEVIEW for Windows on a Single Monitor
Nu-Mega has done it again!
CV/1™ $129
Call Today, Have it Tomorrow

BOUNDS-CHECKER	*$249*
Soft-ICE 2.5	*$386*
MagicCV 3.0	*$199*
CV/1™	*$129*

Special Offer...

Buy BC & S-ICE	*Save $100*
Buy S-ICE & MCV	*Save $86*
Buy all three	*Save $186*

30 Day Money-Back Guarantee

Nu-Mega TECHNOLOGIES INC

CALL TODAY (603) 888-2386 or FAX (603) 888-2465
P.O. BOX 7780 ■ NASHUA, NH ■ 03060-7780 ■ U.S.A.

Circle 4 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Programmer's Journal™

March/April 1991 Volume 9.2

FEATURES

WINDOWS — **WINDOWS 3 INPUT HOOKS** *By Chris Schwartz* . . . 21
Using hook filter functions in Windows 3.0 to enhance your next application

LANGUAGES: C/C++ — **C COMPILERS FOR WINDOWS** *By Paul L. Yao* . . . 28
A comparison of C and C++ compilers that support Windows development

LANGUAGES: FORTRAN — **FORTRAN 90 IS COMING!** *By Thomas M. Lahey* . . . 42
Tom describes some exciting new features incorporated in the Fortran 90 Standard

LANGUAGES: ADA & C — **PRAGMA INTERFACE** *By Richard Riehle* . . . 49
A method for integrating Ada and C with the pragma interface

LANGUAGES: BASIC — **STRING HANDLING** *By Murray L. Lesser* . . . 55
Murray examines how to manipulate BASIC strings using library functions

OS — **PORTING TO THE MAC** *By W. Sumerlin* . . . 60
Some insights and useful recommendations to ease porting an application to Mac

DISASSEMBLY — **DISASSEMBLER ROUNDUP** *By Brett Glass* . . . 66
Brett runs several PC disassembler tools through their paces

LANGUAGES: PASCAL — **EXPLORING XMS** *By Kim Kokkonen* . . . 72
A nifty Turbo Pascal utility that maps XMS high-memory drivers

ASM — **USING EMS FOR FAST FILE I/O** *By John Otken* . . . 75
John integrates EMS for fast file I/O into a real-mode virtual memory system

COLUMNS

PJ FORUM — **AROUND THE BEND AGAIN** *By Dave Thompson* . . . 13

REDIRECTIONS — **PINK NOISE** *By Frank D. Greco* . . . 17

PJ ARMCHAIR — **LANGUAGE LORE** *By M. Steven Baker* . . . 83

CONSULTING — **EFFECTIVE MARKETING PRESENTATIONS** *By William H. Roetzheim* . . . 85

SHORT TAKES — **COBOL REVISITED** *By Kipp Martell and Tim McCormley* . . . 87

IMPRESSIONS — **LANGUAGE PARSERS AND FORTRAN TOOLS** *By M. Steven Baker* . . . 88

OFFRAMP — **CHANGES IN THE FAST LANE** *By Hal W. Hardenbergh* . . . 96

DEPARTMENTS

EDITOR'S NOTE . . . 6

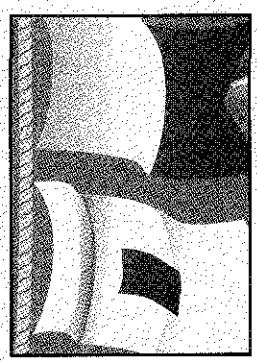
LETTERS . . . 8

PJ PRESSROOM . . . 91

PJ BILLBOARD . . . 94

ADVERTISER INDEX . . . 80

COVER



No matter which language flag you salute, this issue's trove of language lore will help you get your sea legs. The flags on our cover have a hidden message for all of you language-lubbers to decode.

Bon voyage!

Artist: Rob Schuster

Articles and letters in *Programmer's Journal* represent the views of the authors and not necessarily those of the publisher. Letters to the editor become the property of *Programmer's Journal* and may be edited for publication.

Note that all code and listings in *Programmer's Journal* appear "as is," and *Programmer's Journal* makes no guarantee or warranty of their quality, performance, accuracy, or suitability for any particular purpose. *Programmer's Journal* assumes no responsibility or liability to any person or entity with respect to any damage caused or alleged to be caused by their use.

Advertising: For ad rates, deadlines, and placement, call Robert Tappan or Jay Moore at 800-234-0386, or write to *Programmer's Journal*, P.O. Box 70167, Eugene, OR 97401-0110. *Programmer's Journal* is produced using the following software products: Ventura Publisher,™ Aldus PC PageMaker,® and Corel Draw. The PostScript™ files are output on a Linotronic™ 200.

Programmer's Journal (ISSN 0747-5861) is published bimonthly by Oakley Publishing Company, 150 N. 4th Street, Springfield, OR 97477-5454. Second-class postage paid at Springfield, OR, and additional mailing offices. USPS # 000293. **POSTMASTER:** Send address changes to *Programmer's Journal*, PO Box 70387, Eugene, OR 97401-0121

© 1991 OAKLEY PUBLISHING COMPANY

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EDITOR'S NOTE

Misty winter days have returned to the Pacific Northwest, and November's elections have been followed by unfortunate January wars. By the time you read these notes in March, I hope negotiations will have replaced hostilities. I *will* comment on the computer "language wars" so prevalent among both programmers and software vendors. A look in the Letters column of any computer magazine, this one not excepted, will reveal vehement feelings against one language or another. Americans tend to be provincial when it comes to the spoken tongue; fluency in more than one language is uncommon. Maybe this American tendency is evidencing itself in programmers' typically poor attitude toward other computer languages.

CRAZY . . . AFTER ALL THESE LANGUAGES

Microcomputer programming started with hobbyists flipping switches on the front panels of their 8080 boxes. From there, we rapidly evolved to assembly language mnemonics—a big step up from binary. The first popular microlanguage was BASIC (the start of Microsoft's string of successes). Even under CP/M-80's 64K limits, one could find compilers for BASIC, COBOL, FORTRAN, PL/1, Pascal, Modula-2, C, and other, less familiar languages. One advantage of these memory-limited computers is that they forced programmers to design within limits. Better programmers were familiar with the underlying architecture; one's programming would tend to be optimized even when coding in high-level rather than in assembly language.

Programming languages, like other technologies, have life cycles. A new language will have its early proponents, followed by a period of general popularity. The language then goes into decline as another rises to popularity. However, languages differ from other technologies in two ways: a particular strength may make them unique and therefore indispensable, or a large body of existing code representing hundreds of work-years may create inertia and have formidable staying power. A language's success can also depend on compiler availability across platforms, one of C's greatest strengths. Even though C is now dominant, another language is apt to displace it in the nineties. So with computer languages, as with larger issues, negotiations and an open mind are preferable to hostility and defensiveness.

OS/2 LOSERS AND WINDOWS/DOS WINNERS

Fallout continues from the Windows-versus-OS/2 (read Microsoft vs. IBM) debacle. 3COM, once a successful networking vendor, bet on OS/2 and Lan Manager and lost. Because of OS/2's low market sales and the Lan Manager's bugs, 3COM was forced to drop out of the networking software business. Microsoft and Lotus can afford a few marketing mistakes, but a poor decision can be devastating if you're small.

Meanwhile, the DOS extender market continues to thrive. Both Phar Lap and Ergo Computing are introducing 286 DOS extenders designed to work with Microsoft C. Phar Lap's 286|DOS Extender supports both the DOS calling conventions and most OS/2 1.1 API calls (sorry, no Presentation Manager here). You just tell your favorite compiler to generate OS/2 executables and bind in their extender. Ergo's DPMI DOS extender supports both DOS calls and the DOS Protected-Mode Interface (DPMI) built into Windows 3. Binding in a stub for their DOS extender creates a Windows-style EXE file that will run under either Windows 3 or the Ergo DOS extender. SLR's fast OPTLINK/Windows linker is bundled with the Ergo package.

Although Windows is hot, user programs are selling more slowly. If you want to continue writing fast character-mode applications and stay prepared for Windows, MEWEL (Magma Systems) is an interesting product. Quarterdeck will soon release DESQview/X for DOS. Only time will tell which programming targets are the true winners.

Steven Baker

Programmer's Journal

Editor
M. Steven Baker
Managing Editor
Cecelia Hagen
Associate Editors
M. Elizabeth Hardinger
Deborah Healey
Contributing Editors
Michael Abrash, Brett Glass, Frank D. Greco, John Otken, Tom Swan
Founder
Greg Estes

Publisher
Elizabeth Oakley
Associate Publisher
Bobbi Sinyard
Subscriptions
Pamela Coleman
Office Staff
Tina Bates, Barbara Curtright, Jill Edwards, Sylvia Long, Randy Randolph
Production & Art Director
Wanda Walker
Production Assistants
Margie Hanna, Nicole Harris
Illustrations
Jim Carpenter

Sales & Marketing Director
Robert M. Tappan
Advertising Manager
Jay Moore
Advertising Coordinator
Bonnie Chin

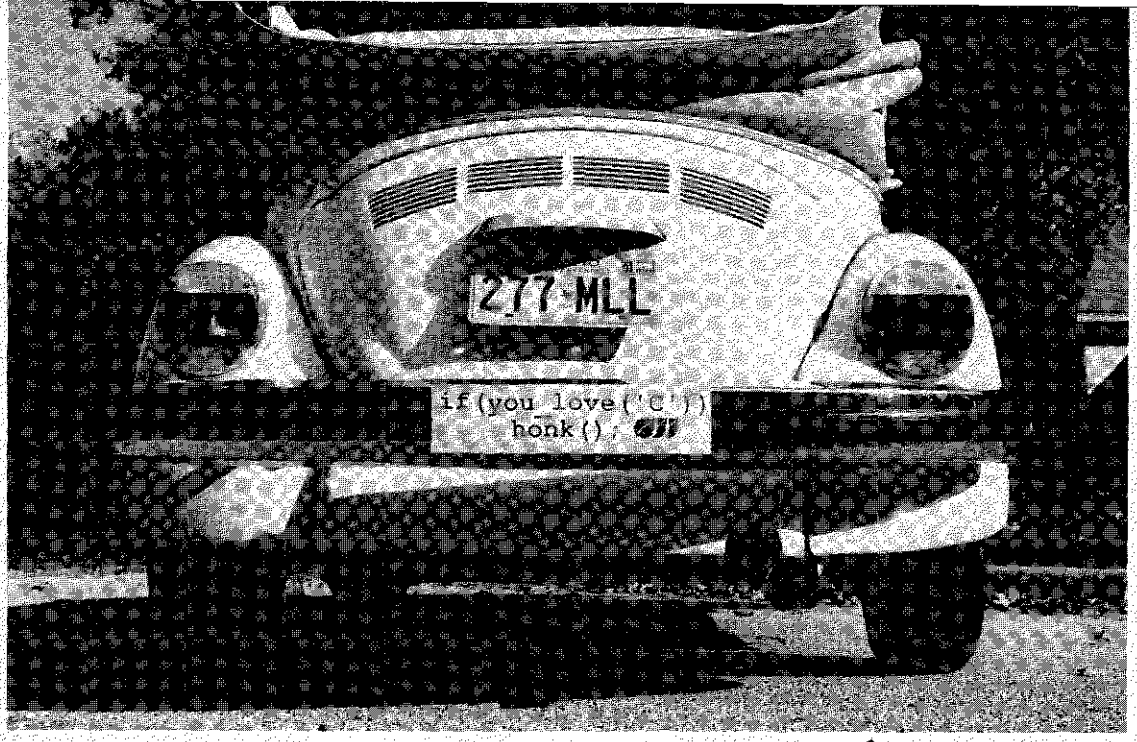
Business and Editorial Offices
Programmer's Journal
P.O. Box 70167
Eugene, OR 97401-0110
(503) 747-0800
(800) 234-0386
Fax: (503) 746-0071
BBS: See page 80

Subscriptions
1 year (6 issues) $24.00

Change of Address
Programmer's Journal
PO Box 70387
Eugene, OR 97401-0121

© 1991 OAKLEY PUBLISHING COMPANY

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



C Programmers Know What Works

Get a Good Start

When you find something that works, you keep using it. When that something is versatile and reliable, you use it a lot. C is that kind of tool. *SuperFunctions* is that kind of library.

Expanded Memory. Pull-Down Menus. Mouse for text and graphics. A Critical Error Handler. These are things you'll need. Maybe you'll be writing a package to do program trading on the Stock Exchange. Maybe your program must decode satellite data on atmospheric ozone.

Whatever it does, every program needs some basics. Like memory management. Like a user interface. You could write these functions yourself, but you don't need to, because they are already in *SuperFunctions*. Don't do what has already been done. Get *SuperFunctions* and get on with your real work!

Expanded Memory

Take along all the luggage you want. If your programs need more than 640k of memory, you can have it with *SuperFunctions*. Sense installed expanded memory. Use the **LIM 4.0** spec to map in pages and pages of expanded memory as data segments. Up to 32 megabytes. Effectively, **memory limitations will disappear** for you.

Critical Error Handler

Get some insurance. The days of letting DOS write "Abort, Retry, Ignore" right on top of your beautiful interface are gone. *SuperFunctions* gives you an elegant means to handle open drive doors, disk, and other critical errors. When a critical error occurs, your program can **intercept the error and recover** using the function in this library.

Logical Windows

Simply a necessity. Every beginner book on C has a chapter on text windows, and that's nice -- for beginners. You'll need more than that. *SuperFunctions* gives you more than 250 logical windows. You can move any one of them to the top with a single function call. You can delete a window and update the screen with a single function call. Write to a window which isn't even on the screen or one that is partially covered by another window. Keyboard idle functions keep the screen alive even when the user is not responding. Write to the window using lots of different text attributes, different colors, different intensities, word wrap on or off, etc. With *SuperFunctions*, it's all simple, automatic, worry free, and fast.

Menus

Use a good map. People use menus because they're simple, useful, and they work. *SuperFunctions* gives you menus that are **simple, useful, workable, and powerful**. One easy to use function determines how big the menu windows should be, displays them, deletes them when finished, accepts first-letter key matches, and indicates the selection to the calling program.

Mouse

Your programs need to know how to control a mouse. *SuperFunctions* gives you that power. **Mouse detection, click detection, position detection.** Complete interface for graphic and text modes.

Extras

SuperFunctions is loaded with extras. But we're running out of room here. We'll just mention that it has **CMOS Timer and Alarm, DOS file functions, advanced time and date, command line parsing,** and more.

Nuts and Bolts

Greenleaf's *SuperFunctions* is a nuts and bolts library with functions you'll really need. It will help you go a long way, and if necessary, you can actually get inside it and fine tune things yourself, because we give you all the source code. **FREE.** So have a ball.

FREE Bumper Sticker

Sure, a bumper sticker is silly. We know that. But it's also kind of fun, and when it's done right, so is programming. Find out more about our libraries and get a FREE bumper sticker, call us.

1-800-523-9830

Ask for extension #200. Also, don't forget to ask about our Gold Support Program.

FAX: (214) 248-7830

SuperFunctions™...$299.

GREENLEAF Software©

Greenleaf Software, Inc.
16479 Dallas Parkway #570
Dallas, TX 75248

Circle 5 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LETTERS

ASSEMBLY VERSUS C

I would like to address the issue of assembly language vs. C program listings in technical publications. The sentiment against assembly language seems to be growing. I am greatly in favor of C language listings for non-hardware-specific algorithms. Recently, however, in technical publications such as *PJ*, I have seen repeated occurrences of hardware-specific, and even op-code-specific, code in high-level languages.

Cutting-edge publications are no place for this kind of garbled code. The advantages of a high-level language (portability and readability) go out the window once you include machine-specific code at the high level. Creating source that has to bend over backwards to get the compiler to perform a specific assembly language instruction is a waste. As Michael Abrash points out in *PJ* 8.6, not only is this code not portable to other machines, it isn't even portable across different versions of the same compiler. Clearly, **int86, outp,** or direct register assignments in a C program listing aren't going to fare too well when compiled on another microprocessor or compiler, or under a different operating system.

By performing register assignments and BIOS and DOS calls directly in your in-line C code, you produce code that is both microprocessor and operating-system dependent. You also generate obscure pseudo-assembly language that is far less readable than a prototyped, C-callable assembly language function—a function that can be clearly documented and can perform exactly the task required. Please, for the sake of clarity and some modicum of object-oriented design, let us keep the C code in C and the assembly code in assembly language.

If you must include such nonsense as **inregs**, **outp**, and **int86**, in-line your C source code, at the very, very least, and proceduralize all such calls into a well-defined set of services, with an appropriate corresponding header file.

Contrary to popular belief, you do not need C++ to do clean object-oriented design. Simple functional decomposition can be accomplished with well-defined services to a set of procedure calls, even if it is only a link library, in any language.

John W. Ratcliff
St. Charles, MO

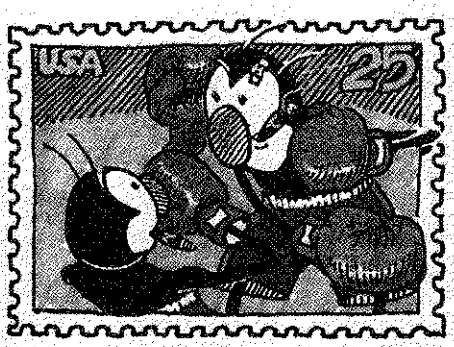


ILLUSTRATION BY JIM CARPENTER

"UNPROTECTED" REDUX

In my "Running Unprotected" article (*PJ* 8.4) on writing a virtual-memory monitor, a few lines of code had the "<" and ">" characters omitted, as well as a "00." Please refer to Listing 1 (p. 12) for the correct lines; they were correct on the code disk and bulletin board listings. I'm also supplying a way to fix a small bug that relates to the adjustment code. The statement **jnz adjust** appears near the end of the article's code. This jump is incorrect because it fails to decrement the loop counter. The correct statement, which redirects the jump toward the end, is also given in Listing 1.

Dean C. Wills
Encino, CA

STOP HUNTING WITH STICKS

In the "Letters" of *PJ* 8.6, James Glass objected to your supposed illustration of Fortran as "Neanderthal." So what? Fortran *is* Neanderthal. Fortran is old technology.

You can put food on the table with a sharpened stick, but your chances of eating are much better with a flint arrowhead. Your odds of eating regularly are better yet if you grow grain and raise livestock and hunt only when the crop fails. I wouldn't argue with Mr. Glass's statements that there are "more total lines of source code devoted to (Fortran)," or "more megabytes of disk storage." There was certainly a time when there were more sharpened sticks than flint arrowheads. But now we consider societies that chip flint as a precursor to food acquisition to be "Neanderthal."

I'm not advocating technology for its own sake, but rather suggesting that a language—or a paradigm—is simply a tool for producing a desired result. I have a friend who makes furniture using only hand tools. His work is awesome, and so are his prices (he estimates he earned $5.50 an hour for my $300 end table). I empathize with my friend's viewpoints. Unfortunately, I can't afford a house full of his furniture. Mostly, I need places to sit, things to cook with, etc., and I need them at a reasonable price. When it comes to software, I also expect it to keep up with hardware advances and changes in the typical user interface. (Do any Fortran programs support Windows 3.0?)

There will never be a perfect language. Our job as programmers is to find the best tool to produce the best program for the problem at hand. A language is just a tool. Use the one that fits, and avoid flint arrowheads if stainless steel is available.

Kevin D. Weeks
Oak Ridge, TN

OFF THE MARK

John Otken is obviously an accomplished assembly language programmer who understands much more about bit tweaking than I do and shares my respect for code readability even at the expense of a few extra bytes or clock ticks. But I take strong exception to a remark in "Optimization Strategies" in *PJ* 8.6.

Otken states that one should "avoid using CS to access data. It is a bad programming practice and almost never necessary." Consider this scenario: assume a system structure that initiates with a load module composed of many low-level support routines and various applications that are called via the MS-DOS **exec** function. Assume further that any application can allocate memory for its own use. When it requires a service from the load module, the application must pass any number of arguments composed of any combination of the registers. Assuming that DS could point anywhere in memory, how best is the far call made? CS is the only segment register whose contents are always immediately known. The most efficient *and* readable coding solution is to set up the far address as a data object in the code segment where it is protected and readily available. Even if the application consists of multiple code segments, the far address can be accessed by calling the original code segment to perform the actual call to the load module. No other registers besides CS and SP

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

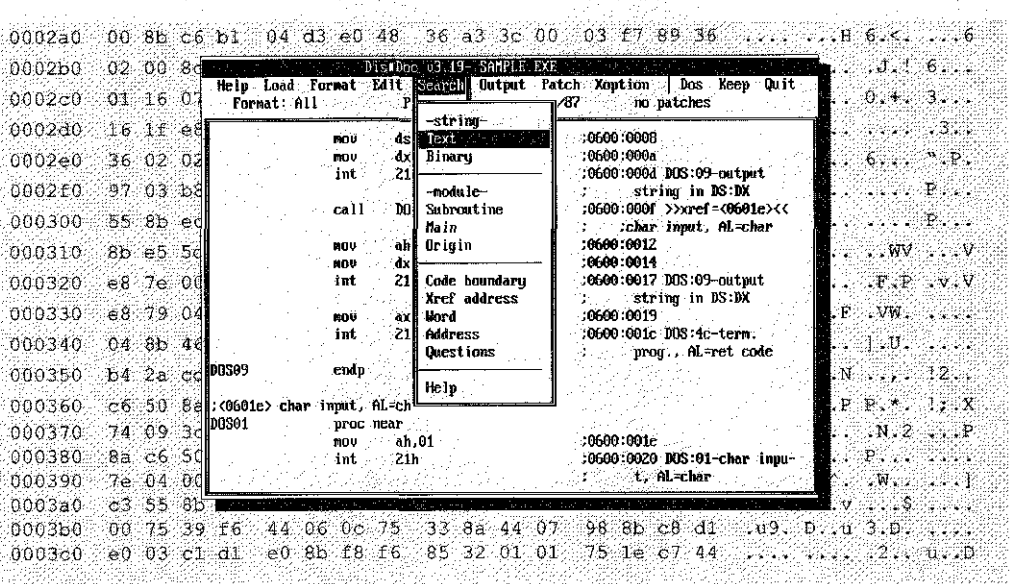
NEW VERSION

No Source?
No Problem.

Automatic BIOS, PSP, MS-DOS interrupt labeling built-in

Batch and Interactive modes

Interactively locates code/data boundaries



8086 to 80486 instruction sets supported

Interactive cross-referencing

Creates listing only after disassembly is perfect

Dis • Doc Professional™

New **Dis • Doc** *Professional* is your **dual-mode** key to any DOS source code. It works in batch and interactive modes simultaneously, allowing you to generate the core information of even the most complex programs **fast**...and modify them even faster. Most programs will come apart in just two minutes. Imagine what you can do with a tool this powerful! **Dis • Doc** sifts through programs **eight times** for guaranteed accuracy. When code gets mixed up with data, our toolbox comes to the rescue with **smart** search and **easy** edit utilities.

Warning: Dis • Doc *Professional* may change the way you work **forever**.

Programmers who used to shy away from fixing outmoded programs with no source code are going to discover a **valuable** new talent: the ability to modify and revise codes that would cost way too much to start over (it's a programming manager's dream). Save your employer huge new-programming fees and enhance your marketability. **Dis • Doc** is so easy to learn, you'll be a high-dollar hero in no time!

Dis • Doc *Professional* is an amazing new teaching tool. Learn how programs work...take them apart and see the writing techniques that top pros use. Use it to assist in debugging. Hunt down viruses and write killers. **Dis • Doc** can save you **years** of frustration, and it only costs $249.95.

COMPARE FOR YOURSELF!

	Dis•Doc™	Sourcer™
Speed of disassembly (bytes/sec)	***822***	*120*
Accuracy (code/data separation*)	***99.9%***	*99.9%*
Unlimited file size	***YES***	*NO*
BIOS labeling	***YES***	*XTRA $$*
BIOS/PSP file labeling	***YES***	*NO*
Unpacks EXE files	***YES***	*XTRA $$*
Patcher	***YES***	*NO*
Batch mode	***YES***	*YES*
Interactive mode	***YES***	*NO*
Time it takes to customize a label	***5 sec.***	*2-5 min.*
Time it takes to change data to code	***5 sec.***	*2-5 min.*

*at eight passes based on a 9040 byte EXE file picked at random

To order your **Dis • Doc Professional™** disassembler toolkit or our $3 demo disk, simply call:

1-800-336-1961 in US
1-203-953-0236 outside US
or send check or money order for **$249.95** plus $5 s&h to:

RJSwantek, Inc.
178 Brookside Road
Newington, CT 06111-1320 USA

For more information call 1-203-953-0236 • FAX available • MasterCard & Visa • Shipped Immediately Via UPS Blue inside US.

Sourcer is a registered trademark of V Communications, Inc.

Circle 6 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

are used. When the call returns, even these are painlessly restored—the process is simplicity itself.

Maintaining the pointer in DS while observing system functionality would require passing the destination DS as yet another argument to the service routine (thereby complicating it and the general interface, again muddying readability). This would require constant DS switching. Other possibilities, such as accessing the load module via the interrupt table, could lead to portability problems. No, the best solution is to keep the pointer in CS.

I would say that Otken is correct in his statement that CS is not appropriate for storing or accessing data, and he is probably correct that it is "almost never necessary." But declaring it a "bad programming practice" in an article that seems to stress readability as well as optimization is off the mark. Come on, John. Haven't you ever used a screwdriver as a chisel when its blade is just the right size for the job?

Michael L. Giancola
Pacific Palisades, CA

JOHN OTKEN RESPONDS: The 8088 can make a far indirect CALL or JUMP through one of four segment registers. The best choice depends on the program. Use the first one of DS, SS, or ES that points to DGROUP. If none of them does, load DS or ES with DGROUP, and then CALL through that register. If that doesn't work, PUSH values on the stack and RET. If that code looks gross, use a CS override. A CS override may offer the best solution to a problem, but that alone is not enough to elevate it above the pit of "bad programming." Using DS:DX as a string pointer, for example, is a very bad programming practice.

WRITING IN THE SAND

I have a question. I can't allocate memory with EXE programs written in my assembly language. I remember John Otken referring to a raw-memory allocation routine in a *PJ* article, and I think that routine might be what I need.

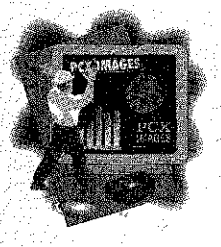
I have no problem if I'm allocating-deallocating memory with COM programs, but when I try to do it with EXE programs, DOS tells me that there's not enough memory or that there are only three paragraphs of memory left, which is not true. My programs become huge when, for example, I have to declare so many fixed-memory segments for video buffers. I would appreciate your help.

John Kibedi
Vancouver, B.C., Canada

JOHN OTKEN RESPONDS: When DOS loads an EXE file into a memory block, it allocates the whole block. When your program starts, it already owns most of memory and cannot allocate any more. Shrink the program memory block so that you can reallocate memory. Listing 2 (p. 12) is a small demonstration program; for more information, see my column in PJ *8.1.* PJ

ERRATUM: The wrong telephone number was published for Mike Schmit in his article in *PJ* 9.1. The correct number is (408) 244-6826.

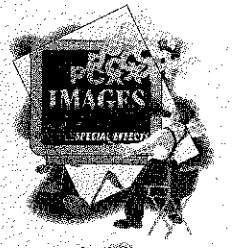
ADD THE POWER OF GRAPHICS TO YOUR PROGRAMS WITH GENUS



PCX PROGRAMMER'S TOOLKIT 4.0

Finally, a programmer's toolkit that helps you incorporate graphics into your programs quickly and easily. The PCX Programmer's Toolkit includes over 75 routines to display, save, scale, and print PCX bitmapped graphics from almost any program.

ONLY $249



PCX EFFECTS*

Add spectacular special effects (F/X) to your programs, without being trapped in someone else's editor or slideshow program. Wipe, split, crush, slide, sand, drip, diagonal, spiral, random, or explode your graphics.

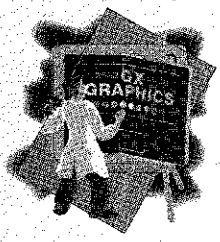
ONLY $99

* Requires PCX Toolkit

PCX TEXT

Now you can display blazing bitmapped text in any graphics mode as simply as text mode. With PCX Text, you can display your textual information right along with your graphical data.

ONLY $149



GX GRAPHICS

A complete graphics library supporting all graphics primitives. Use the GX Graphics toolkit instead of the BGI or MS libraries and make your program faster, smaller, and more portable across compilers – while supporting more video modes.

ONLY $199

FOR THE GENUS GX DEVELOPMENT SERIES CATALOG, CALL TOLL-FREE:

1-800-227-0918

VISA, MASTERCARD, AMERICAN EXPRESS, CHECK, C.O.D., PURCHASE ORDER

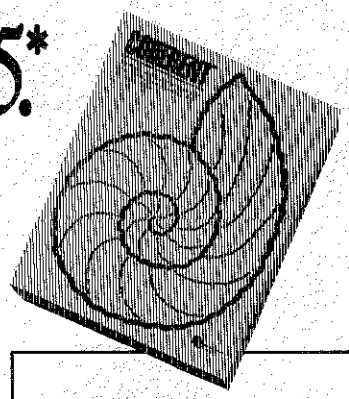
G·E·N·U·S MICROPROGRAMMING

11315 MEADOW LAKE • HOUSTON, TX 77077 • (713) 870-0737
FAX (713) 870-0288 • BBS (713) 266-9362

Circle 7 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Everything You Ever Wanted In UNIX. And Less. $99.95.*



OK. We know it's hard to believe. So just consider this. Coherent™ is a virtual clone of UNIX. But it was developed independently by Mark Williams Company. Which means we don't pay hundreds of dollars per copy in licensing fees.

What's more, Coherent embodies the original tenet of UNIX: small is beautiful. This simple fact leads to a whole host of both cost and performance advantages for Coherent. So read on, because there's a lot more to Coherent than its price.

SMALLER, FASTER...BETTER.

Everybody appreciates a good deal. But what is it that makes small so great?

For one thing, Coherent gives you UNIX capabilities on a machine you can actually afford. Requiring only 10 megabytes of disk space, Coherent can reside with DOS. So you can keep all your DOS applications and move up to Coherent. You can also have it running faster, learn it faster and get faster overall performance. All because Coherent is small. Sounds beautiful, doesn't it?

But small wouldn't be so great if it didn't do the job it was meant to do.

LESS IS MORE!	Coherent For the IBM-PC/AT and compatible 286 or 386 based machines.	Santa Cruz Operation's XENIX 286, Version 2.3.2
No. of Manuals	1	8
No. of Disks	4	21
Kernel Size	64K	198K
Install Time	20-30 min.	3-4 hours
Suggested Disk Space	10 meg	30 meg
Min. Memory Required	640K	1-2 meg
Performance*	38.7 sec	100.3 sec
Price	$99.95	$1495.00

*Byte Exec benchmark, 1000 iterations on 20 MHZ 386. Hardware requirements: 1.2 meg 5¼" or 1.4 meg 3½" floppy, and hard disk. Does not run on Microchannel machines.

EVERYTHING UNIX WAS MEANT TO DO.

Like the original UNIX, Coherent is a powerful multi-user, multi-tasking development system. With a complete UNIX-compatible kernel which makes a vast world of UNIX software available including over a gigabyte of public domain software.

Coherent also comes with Lex and Yacc, a complete C compiler and a full set of nearly 200 UNIX commands including text processing, program development, administrative and maintenance commands plus UUCP.

CRITICS AGREE: IT'S AN INCREDIBLE VALUE!

"Mark Williams Co. seems to have mastered the art of illusion; Coherent comes so fully qualified as a UNIX clone, you find yourself thinking 'I can't believe it's not UNIX.' "
–Sean Fulton, UNIX Today!, November 26, 1990

"...(Coherent) may be the best thing that has happened to UNIX yet."
–William Zachmann, PC Week, November 5, 1990

"If you want to come as close as you can to real UNIX for a low price, COHERENT can't be beat."
–Warren Keuffel, Computer Language Magazine, November 1990

"If you want a UNIX-like development and learning system for less than $100...I don't see how you can go wrong with Coherent."
–David Fiedler, BYTE Magazine, November 1990

EXPERIENCE, SUPPORT AND A 60-DAY MONEY BACK GUARANTEE.

Wondering how something as good as Coherent could come from nowhere? Well it didn't. It came from Mark Williams Company, people who've developed C compilers for DEC, Intel, Wang and thousands of professional programmers.

NEW COHERENT RELEASE 3.1
NOW WITH...
–elvis: vi editor clone
–SCSI (Adaptec AHA 154x series and more on the way.) and ESDI support
–UUCP Bulletin Board System
–RAM disk support
–And much, much more!

OVER 10,000 SATISFIED USERS!

We make all this experience available to users through complete technical support via telephone. And from the original system developers, too!

Yes, we know $99.95 may still be hard to believe. But we've made it fool-proof to find out for yourself. With a 60-day money-back no-hassles guarantee.

You have to be more than just a little curious about Coherent by now. So why not just do it? Pick up that phone and order today.

You'll be on your way to having everything you ever wanted in UNIX. And for a lot less than you ever expected.

1-800-MARK WMS
(1-800-627-5967 or 1-708-291-6700)
FAX: 1-708-291-6750
60-DAY MONEY BACK GUARANTEE!

Mark Williams Company
60 Revere Drive
Northbrook, IL 60062

*Plus shipping and handling. Coherent is a trademark of Mark Williams Company. UNIX is a trademark of AT&T. XENIX is a trademark of Microsoft.

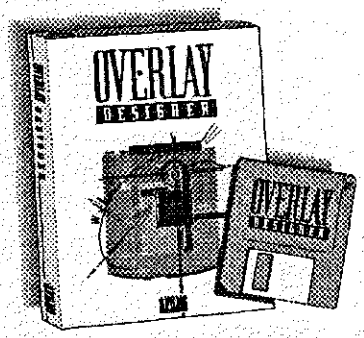
Contact this advertiser directly.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

OVERLAY DESIGNER™

for use with

Plink86®and RTLink®



IF YOUR PROJECT BOGS DOWN

in the tedious detail work involved in designing efficient overlay structures, Overlay Designer is the software tool to save you time and keep your project on schedule.

DO NOT BE MISLED ...

by products that claim to do overlays automatically for large DOS applications. Important tradeoffs between size and performance still must be made by programmers who understand the code and the user interface. The powerful ***"what you see is what you linked"*** interactive Overlay Designer editor, tabulation, and graphical modeling program is the only workable solution. Early adopters of Overlay Designer use it regularly and find it an ***indispensible tool for overlay design.*** Context sensitive help and automatic .MAP and .OBJ file reading give immediate results with ***virtually no set-up required.***

"After my first day with Overlay Designer, I had done work that would have otherwise taken me 2 weeks."
- Mike Milsner, Senior Programmer
Symantec Corporation

Overlay Designer is backed by a 30 day unconditional money back guarantee.

COLEMAN SOFTLABS INC

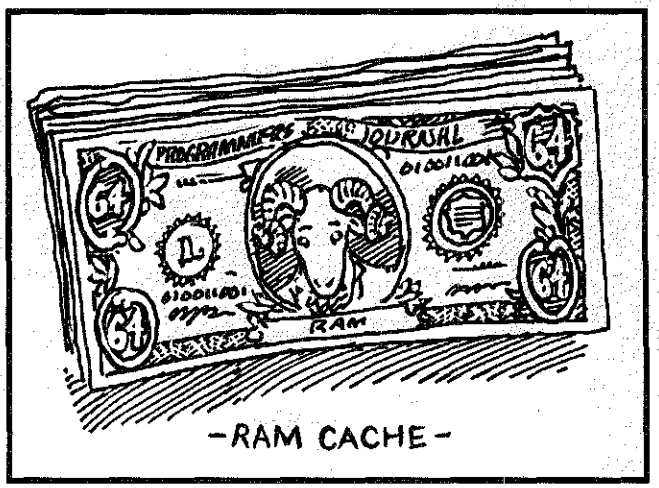
Coleman Softlabs Inc.,
296 Bay Road,
Atherton, CA 94027.
(415) 322-9006 phone
(415) 322-1622 fax

LISTING 1

```
;Corrections to Dean Wills' code in "Running Unprotected" in PJ 8.4

idt     LABEL   BYTE                    ;INTERRUPT DESCRIPTOR TABLE
IRP     z,<00,01,02,03,04,05,06,07,08,09,10,11,12,13,14,??>
        DW      OFFSET _TEXT:fault&z, x_TEXT,(XINT_GATE OR 80h) SHL 8,0
ENDM
;       17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
IRP     z,<16,??,??,??,??,??,??,??,??,??,??,??,??,??,??>
        DW      OFFSET _TEXT:fault&z, x_TEXT,(XINT_GATE OR 80h) SHL 8,0
ENDM
IRP     z,<00,01,02,03,04,05,06,07,08,09,0a,0b,0c,0d,0e,0f>
        DW      OFFSET _TEXT:inthw&z, x_TEXT,(XINT_GATE OR 80h) SHL 8,0
ENDM

;  Code to redirect jump for Dean Wills' "Running Unprotected"

;;;; comment out: jnz   adjust      ;no, must be a gate- no adjustment
        jnz   $next       ;jump to end of loop
        ...
        ...
        ...
        stosw
 $next: loop  adjust      ;loop
        ret
_TEXT   ENDS
```

LISTING 2

```
; Code from John Otken

    dosseg
    .model small
    .stack
    .code

;;  main
;
main proc
    mov  bx,offset stack+15 ; compute program size in paragraphs
    mov  cl,4               ; SS is DGROUP in program
    shr  bx,cl
    add  bx,data
    mov  ax,es              ; ES -- Program Segment Prefix
    sub  bx,ax

    mov  ah,4Ah             ; shrink program's memory block
    int 21h

    mov  ah,48h             ; allocate 16 paragraphs of memory
    mov  bx,16              ; (16 paragraphs == 256 bytes)
    int  21h

    mov ax,4C00h            ; exit program
    int  21h
main endp

    end main
```



Circle 8 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Around the bend again

By Dave Thompson

ILLUSTRATION BY JIM CARPENTER

The rumors of my death have been greatly exaggerated (to quote Mark Twain or John Wayne or someone equally settled). The rumors began when I announced that I was closing down *Micro Cornucopia*. Closing down *Micro C* has been interesting—it's been more work, more trauma, and a lot more upheaval than I expected. Am I glad I started *Micro C*? Yes. Am I glad I shut it down? Yes again. But it wasn't easy.

Micro Cornucopia was a magazine that focused almost exclusively on computers, airplanes, fractals, and banana bread. It was *Mad*, *Mother Earth News*, and *The Hackers' Home Companion*. (If you subscribed to *Micro C*, then you know what it really was.) Anyway, I published the fifty-third and final issue on April 1, 1990.

There's something about spending nine years focused intensely on one thing. Eventually a time comes when all those other things that you tell yourself you'll get around to quit nagging at you and picking at you and start whumpin' you over the head. Toward the end, I'd gotten a pretty good case of scalp divots.

I've been bicycling (spent a week riding across Oregon), skiing, workshopping (very different from mall-shopping), flying, hiking, and finding out what can be done with great gobs of unplanned time.

LITE PROMOTION

One of the most . . . uhm . . . innovative people I know is Fred Cisin of Xenosoft. He's written a couple of small software packages and he markets them on a mostly/partly full-time basis. I mean, he's sometimes in the office and sometimes not.

He does go to a few shows—West Coast Computer Faire, Fall Comdex—you know, the sort of shows that one-man operations used to go to before IBM and Apple and NorthGate and Epson and NEC discovered computers.

I saw him at Comdex the last time I was there. Smallest booth in the show. Pretty easy to miss. However, Fred had a magic marketing weapon, a weapon so powerful that it kept his tiny operation going, paying all those Las Vegas hotel bills and footing the bill for the booth space and the evenings on the town to boot.

His secret weapon—beer.

There's nothing like a cold beer, a really *cold* beer, when you've been on your feet all day. And if you're a famous computer columnist and you make it a habit to stop by the Xenosoft booth for a cold one and while you're there Fred happens to hand you a copy of his latest product . . . now that's promotion. (If you're curious about the results, you might check page 64 of the November 26 issue of *Infoworld*.)

DEFINITELY NOT WHITE NOISE

The one thing I really liked about my old CP/M machine (it's a Big Board, for those of you who've washed; for those of you who haven't, it's the father of the Xerox 820 and the Kaypro) was its silence. Once that beast had booted and those 8-inch Shugarts had ground to a halt, it was just me and the Z80.

As clones invaded our shores and I moved on to XTs and ATs and 386s, things got a lot noisier. The fans in those early power supplies sounded like Isuzus running motocross. And the 5-meg full-height hard drives? During windup they sounded so much like jet engines I often thought about sneaking them out to Bend's airstrip to give the hangar jocks a thrill.

"Hey Joe, does that sound like a turbine?"

"Yep, a big one."

"Maybe two."

"Might be a DC-8. Noisy, those 8s."

"Spooling up in the restroom, sounds like."

"Definitely an 8."

So I've been looking for quiet. I do my best stuff at night, and I'd prefer not having a silly mechanical device keeping me awake while I work.

Plus I've heard that noise, even the high-pitched whistles from monitors, causes tension. It certainly makes me tense. So I changed to a higher-resolution monitor, which moved the whistle (horizontal oscillator) up above my hearing range. Then I went after the low-frequency buzz.

SILENCE IS GOLDEN

I slowed down the power supply fan by adding a resistor to its supply lead. Cutting the blade speed in half cut the noise by a factor of ten. I also dug around for the quietest 3½-inch hard drive on the market, and I made it even quieter by unplugging its power cable. I hook it up when I need something. Handy, these flip-top cabinets.

I'm also using a Toshiba T1000 for notes and stuff. The T1000s are cheap, they're silent, and the ones with the vanilla LCD screen (not the lit chocolate-chip screen) will run five hours on a charge. If you replace the four sub-C nicads with four C nicads you can get ten hours on a charge.

Just open up the system and remove the plastic assembly that holds the small batteries. Then wire four C cells in series

Dave Thompson is a former hardware/software engineer at Tektronix and the past editor of the prehistoric programming magazine Micro Cornucopia. *He can be reached at P.O. Box 223, Bend, OR 97709.*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

and solder the red lead from the chassis to the "plus" end. Then wrap the C cells in insulating tape so they make a nice flat bundle, and stick the bundle between the motherboard and the cabinet (right where the plastic assembly was) with sticky-backed Velcro. It's not hard. Save the old batteries—you'll have to reinstall them if you need to return the machine under warranty.

PREDICTING THE FUTURE

I don't usually try to predict the future; I used to, but I quit after people started remembering what I'd predicted. However, I'm so certain about this prediction that I've got to tell you—so you can remember and tell me I was right.

Two years from now I expect laptops to make up at least 80% of PC sales. I don't expect them to be used mainly as traveling machines, either—they'll spend most of their time on office desks. But they'll be cheaper than today's standard boxes (6 lbs. of material instead of 50), and they'll take up less desk space. They'll also be just as fast, have the same resolution color videos, and they'll have spacious hard drives. Executives won't be taking home floppies or removable hard drives to do their evening work; they'll simply briefcase their machines.

There are three reasons it hasn't happened yet. First, flat screens are just now showing up in color and in high resolution. Second, manufacturers are working out the problems of packing 40, 60, 80, or 100 megs on a 2½-inch drive. And third, the CMOS versions of the faster chip sets are just becoming cheap.

Buying a new system? Wait. They'll be getting a lot cheaper as AMD and others hit the market with 386-compatible processors. That should sharply reduce the price of 386 boards. And if you paste together a 386 with a 387 (you can already get clone 387s) you've got most of a 486, so cheap 386s (thrills) will no doubt put pressure on Intel to cut the price of the 486. (I hope you're not working for someone whose product line is tightly harnessed to the 286.)

The move to laptops should also depress the price of standard configurations. As prices of laptops come down, you'll see the manufacturers of the big systems holding fire sales.

MAKE YOURSELF INVALUABLE

With software companies shifting their emphasis to Windows, I see no need for competent programmers to drop anchor or batten down the hatches because of economic storm warnings. If you C well enough to do Windows or do Windows well enough to C, it's clear that your visibility is unlimited. If you can endure the pane you'll make your mark on the screen of civilization. I get GUI just thinking about it. Oxymoron for the day: Microsoft Works. (And you wonder why editors get glassy-eyed when they see my stuff. I promise, no more puns or cliches or humor—till next time.)

The Programmer's Corner
— A Source Code Repository —

If you are reading this magazine, then you appreciate the art of programming. One of the most important tools for a software craftsman is example code. To meet this need, we provide **more than 9000 files** (over one gigabyte) of shareware and public domain source code and programs. We cover all popular languages including C, Pascal, and Basic. TPC even has hard to find source code such as OS/2 PM C source, C++ source, and Turbo Pascal 5.5 OOP source. Source code from this issue of *Programmer's Journal* is also available. New code is added daily, all in the interest of helping the professional programmer.

So call today: (301) 596-1180
(301) 995-3744
(301) 621-4636
300-9600 Baud
24 Hours-A-Day

Only $50 for a one year subscription

— TPC uses FastComm 9600 V.32 Modems —

TPC
The Programmer's Corner
5430 F Lynx Lane, Suite 224
Columbia, MD 21044

Circle 10 on Reader Service Card

Data Junction®
converts data files!

Any ASCII • dBASE • R:BASE • Lotus 1-2-3
Excel • Paradox • Informix • Q&A • Oracle • Clarion
DataFlex • askSam • SC4 • Xdb • DataEase • Symphony
ACT! • Platinum • DAC • ACCPAC+ and other apps
Btrieve • c-tree • EBCDIC • Packed & Zoned decimal
IBM/Mips/Sparc/Cray/Vax/Intel/68000 Binaries & Floats

Filter Your Files While Converting!
- *Menu driven — No programming*
- *Select & rearrange records/fields/chars*
- *Instant scripting for batch converts*
- *Written in C — Ideal for OEM apps*

Clean Up & Massage Your Data!
- *Merge & split fields; Name/address parsing*
- *Search/replace and case translation by column*
- *Edit/enter corrections — Build custom data types*

1990 DATA BASED ADVISOR The Database Management Systems Magazine — Readers Choice Award

"... an invaluable integration tool." —*Byte*
"... more than worth its price." —*PC Magazine*
"... a surefire winner." —*Computer Language*

Standard $99—Converts dBASE, ASCII, 1-2-3, Q&A, Enable, DIF, askSam, & merge files for WordPerfect, Word & Wordstar
Professional $199—Standard formats plus Binary/EBCDIC, SC4, SYLK, Paradox, R:BASE, Clarion, Excel, DataEase, DataFlex, c-tree, Btrieve
Advanced $299—Standard & Prof. formats + Informix, Xdb, Oracle, C-ISAM

Call **800/444-1945** or 512/482-0824 • FAX 512/482-0976
Also available for LANs, Xenix & Unix. MC/VISA • AMEX • COD • Corp. PO.
Tools & Techniques Inc • 1620 W 12 • Austin TX 78703

Circle 11 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The MetaWare Extended-DOS High C ® *Globally Optimizing Compiler* is for developing mission-critical applications that require:

- Compilation of large programs (one million+ lines)
- Ease of conversion of Microsoft "or other" code to 32-bits
- Support for the MetaWare 32-Bit Source-Level Debugger
- 100% ANSI Conformance
- Ease of migration to other development environments: DOS, UNIX System Vr3&4, SunOS, AIX, ...
- Global optimizations and Profiler for maximizing speed of code
- C++ as a future development language

The MetaWare Extended-DOS High C *Locally Optimizing Compiler* extends the capability of our industry-proven technology by:

- Exploiting the Power of AutoCAD® ADS and DESQview/X™
- Ease of conversion of Microsoft "or other" code to 32-bits
- Providing an execution Profiler that helps developers increase execution speed by identifying bottlenecks
- Supporting the MetaWare 32-Bit Source-Level Debugger

Professional software developers live and breathe in challenging development environments unknown to most programmers. As a matter of survival, these Pioneers depend on the precision, performance, and portability that only MetaWare delivers. The superior reliability and functionality of our quality compilers and tools allow these Explorers to navigate unfamiliar development terrain with confidence.

But even the best tools are not enough. You need a close, professional relationship with the veteran team that pioneered Extended-DOS compilers way back in 1986; this technical support is the best "Life support" available today. MetaWare now extends your development horizons with two choices in compiler technology. Both *Locally* and *Globally* optimized technologies support the new MetaWare 32-bit Source-Level Debugger.

To learn more about how MetaWare can help you survive in your environment, or to place an order, call now.

MetaWare®
INCORPORATED

Compiler Products for Professional Software Developers

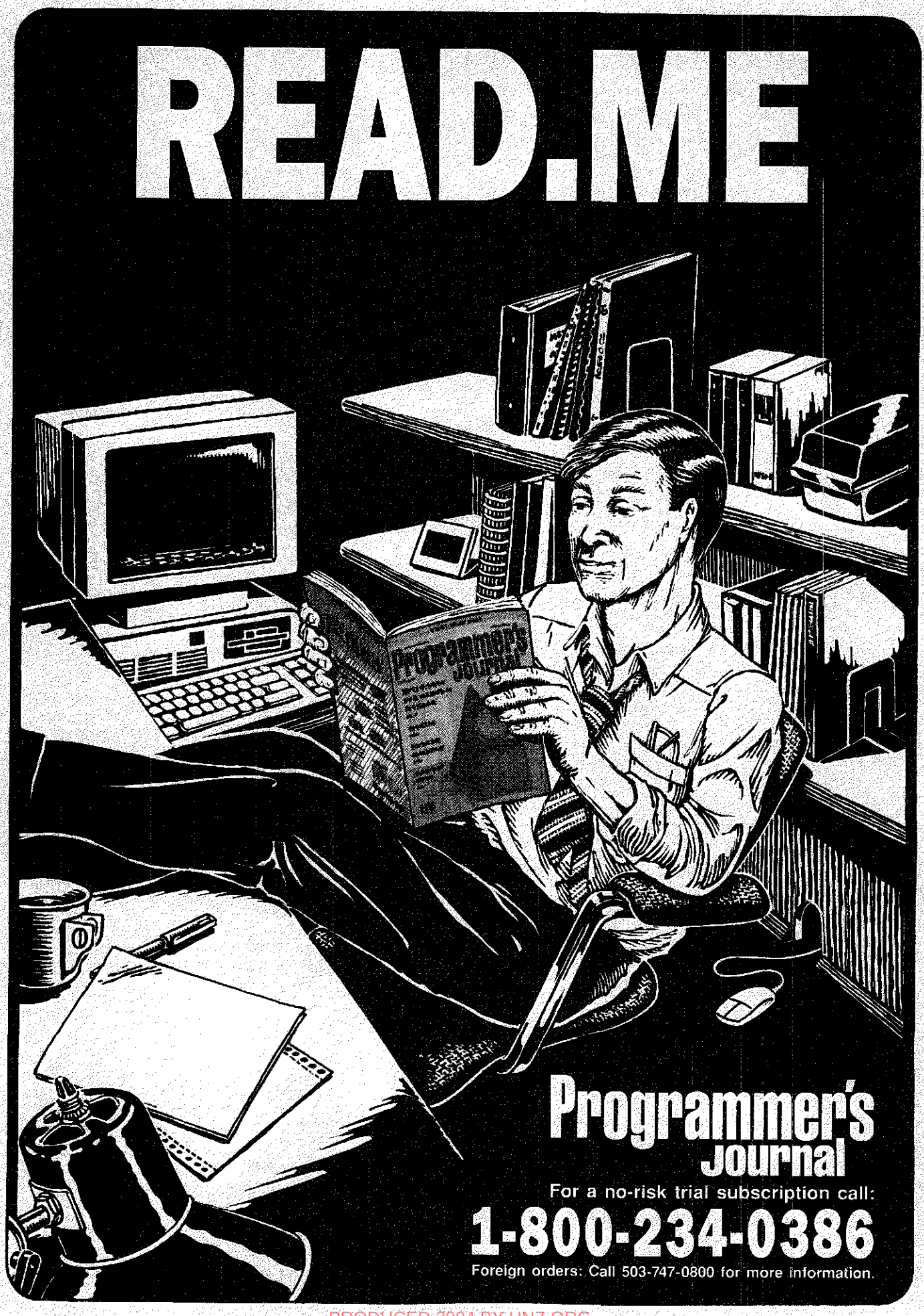
Circle **12** on Reader Service Card

2161 Delaware Avenue • Santa Cruz, CA • 95060-5706 • 408/429-6382 • FAX 408/429-9273

MetaWare, High C, and Professional Pascal are registered trademarks of MetaWare Incorporated. Other names are trademarks of their respective companies.
© Copyright 1991 MetaWare Incorporated

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

READ.ME



Programmer's Journal

For a no-risk trial subscription call:

1-800-234-0386

Foreign orders: Call 503-747-0800 for more information.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Pink Noise

By Frank D. Greco

Solarix Systems has entered the SPARCstation-compatible market with what they call a "truly scalable" SPARC workstation. According to a marketing representative I spoke to at the recent Sun User Group conference, the new $6,995 Solarix/4 PW+ starts out at a leisurely 18 MIPS and can be upgraded in performance by merely swapping in a credit-card-sized module called the A-Module. Solarix is promising that a 40-MIPS upgrade will be available in late 1991. This workstation is unlike other SPARCstation clones in that the A-Module uses the 64-bit MBUS (not the SBUS) and some electronic-timing trickery for simple processor upgrades. The SPARC industry standard SBUS is also there for adding Sun-compatible peripherals.

Toshiba, Fujitsu, Xerox, Matsushita, CompuAdd, Tatung, Goldstar, and Hyundai have all entered the high-growth SPARC arena. Rumors have it that other players such as Acer, Dell, or AST may now join in the fray.

Even unflappable *PJ* Editor Steve Baker and the inimitable Hal Hardenbergh are now using Sun workstations.

OS/2 NT IS NG FOR COMPAQ

Microsoft is loudly whispering that the New Technology version of OS/2 (OS/2 NT) is being targeted for MIPS's RISC microprocessors. Apparently they hope to lure manufacturers like Compaq into using OS/2 on a fast platform. Compaq has been making noises for a while now about jumping on the RISC bandwagon for the upcoming wave of high-speed commodity workstations. Compaq could be merely jockeying for position with Intel because they don't enjoy seeing Intel, as a vendor and supplier, selling

ILLUSTRATION BY JIM CARPENTER

competing 386/486 Unix workstations. Or they could be legitimately looking to choose a viable workstation vendor.

COMPAQ TO REALITY

Rod Canion's crew has reduced their RISC choice to either the SPARC or MIPS microprocessors. Frankly, I don't see the benefit for Compaq in using the MIPS architecture. There is already a fast-growing market for SPARC hardware and software—Compaq would be foolish to ignore this potential. The MIPS chip does not have a substantial software industry growing up around it, or even a moderate one. This is an essential ingredient for any successful computer product line. Witness the MIPS-based DEC and Sony workstations (an unusually bad business decision on Sony's part)—with barely a trickle of commercial software. Even if Microsoft does port OS/2 to the MIPS architecture (which we predicted here in January 1987), who cares? The computing community doesn't care, and I don't believe Compaq does either.

TOO HOT NEAR THE SUN?

The only reason that Compaq may not choose the SPARC architecture would be the stiff competition from Sun. But it wouldn't be the first time that Compaq took on a pacesetting incumbent. Compaq is a big winner in the IBM-generated personal computer field; there is no reason why they can't also be successful in the Unix/SPARC market. The substantial amount of SPARC-based shrink-wrapped software will continue to expand, and Compaq's customers would have access to all this new software. Let's face it—the volume of available software for the MIPS processor won't even come close to the SPARC software market.

COMPAQ COULD GO TO MARS

Compaq could create a machine similar to the Mars Microsystems workstation. Along with its 15-MIPS SPARC processor running SunOS and OpenWindows, the MM workstation has an additional 80386 processor to run MS-DOS/Windows at their full speed.

If Compaq chooses the MIPS architecture for their next-generation machines, it will be a big mistake. The upcoming commodity workstation market will be based on SPARC, not on any other processor.

SUN AND HDTV

It seems to me that Sun is after something much bigger than the workstation market. Why did Sun bring together the Macintosh guru Andy Hertzfeld, the master industrial designer Hartmut Esslinger of Frogdesign (designer of the NeXT and SPARCstation packaging), and the hardware guru (and Sun founder) Andreas Bechtolsheim to develop a SPARC-based television? Yes, you heard right. Andy Bechtolsheim shrugged off my questions at the SUG conference, saying only that the threesome was working on a big government contract. I don't think so.

COPERNICUS WOULD BE PROUD

Imagine a super-intelligent SPARC-powered interface to HDTV video and digital audio equipment that would allow nonexpert users a truly easy way to use increasingly sophisticated electronic gadgetry. How about extending OPENLOOK's drag-and-drop metaphor

Frank D. Greco is an independent consultant who provides industry analysis, systems consulting and technological forecasting for Fortune 500 companies. Interested readers may get in contact with Frank c/o Crossroads Technologies, P.O. Box 530, Fanwood, New Jersey 07023, (201) 754-7820.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

to VCR programming? You see an on-screen TV program listing, and you use the little hand-pointer glyph to grab an interesting show:

7:00 p.m.—Mr. Ed: Wilbur takes Mr. Ed on a tour of the glue factory.

11:30 p.m.—Star Trek: Spock jams with hippie aliens in the twenty-third century.

4:30 a.m.—Movie: Beer and Loafing in New Jersey.

Drag one of these onto an iconic representation of your VCR and bingo—your VCR is now programmed. Beats trying to decipher a bad translation of some Japanese VCR documentation.

Or how about some software that senses the signals from the major networks that alert affiliate stations when a commercial is pending and enables your VCR to skip it (imagine the controversy!).

LAND OF THE RISING SUNS

The giant Matsushita Electric Industrial of Japan has announced several powerful computers based on SPARC. Matsushita plans to incorporate the SPARC processor in consumer electronics products—probably in the upcoming huge HDTV market. Will Sun use the Hertzfeld, Esslinger, and Bechtolsheim system created for the "government bid" that Bechtolsheim alluded to? Of course they will.

Does the recent Matsushita purchase of MCA bode well for Sun and their SPARC architecture? Matsushita has the consumer electronics expertise (and money), MCA has tremendous clout in Hollywood (the video "software"), and Sun has the technical expertise. The market for computers, televisions, and ISDN is much, much bigger than the workstation market but will grow very slowly until about five years from now. Then watch it explode.

It seems Sun is carefully positioning themselves for this market. Borrowing a page from the Japanese, who of course borrowed it from us (said in my very best John Wayne voice), Sun is focusing some research toward the long term. As long as they don't lose their short-term focus in Unix-based computer workstations, Sun can come off a huge winner.

LAPS OF LUXURY

Joining Toshiba, Fujitsu, and Matsushita in the SPARC laptop field is an interesting 15-MIPS/1.6-MFLOPS entry from a California company called Research, Development, and Innovations (RDI). The SLT-200 laptop runs industry-standard SPARC software that is fully compatible with Sun's line of workstations (SunOS with System V Release 4 enhancements). It also runs PC-based binaries with the assistance of SoftPC, an excellent pure-software PC emulator, and Macintosh programs. RDI claims to have built software that traps Macintosh system calls and translates them into Unix or OpenWindows calls at the speed of a Macintosh SE.

The SLT-200 comes with 8 megabytes (8 megabytes is to a workstation what 640 kilobytes is to a PC) and is expandable to 32 megabytes. It also has a 28-msec 120MB disk, an Ethernet interface, and a back-lit super-twist monochrome LCD with a resolution of 1152×900. For all that, it only weighs about five pounds with batteries. Prices range from $7,000 to $12,000.

By the way, you should check out SoftPC running Microsoft Windows 3.0 on the new 29-MIPS SPARCstation 2. PC applications in this completely emulated environment are usually faster than native applications on an 80386. SoftPC copies the hardware of the PC down to the register level; with emulation this thorough, you have no worry about handling register incompatibilities, undocumented DOS calls, applications doing direct video memory-mapping, and so on.

For those of you who shun emulation, Puzzle Systems of California has an interesting SBUS card alternative for your SPARC-based workstation. Their 80386 microprocessor runs your favorite Windows 3.0 applications at full speed and integrates within Sun's windowing system so you can cut and paste among Windows/DOS and Sun applications.

OS/2 IS DEAD

Now that Microsoft has handed over future development of OS/2 to IBM and turned their attention to Windows, we can safely say that OS/2 is dead. Microsoft will turn Windows into what OS/2 should have been (and what GeoWorks is). OS/2 and Presentation Manager from IBM will be forgettable. Clearly, Gates and Company will slowly transform Windows into more than the user interface. It's the old "end around" strategy.

IT'S ALL IN THE CARDS

A few "Redirections" ago I mentioned that Canon is using credit-card-sized optical storage devices that can hold up to 10 megabytes. These rectilinear optical ROM devices are licensed from technology patented by a company called Drexler. Now another company, MicroMirror Corporation of Colorado Springs, has created similarly sized devices (but about twice as thick) using a technology based on the reflectivity of the optical material—essentially, tiny mirrors. They claim their first-generation versions can store 85 megabytes! And they speculate that when commercial diode laser technology becomes inexpensive enough (probably five years from now), they'll be creating credit-card-sized devices that can store 850 megabytes of data.

FIELD OF STREAMS

All of Sun's BSD-flavored networking API (i.e., sockets) are now layered on top of the cleaner, stratified AT&T System V-based Streams drivers. Soon, BSD-based Unix pipes will be based on the Transport Layer Interface (TLI) in System V Release 4 instead of on sockets. This means that Unix pipes themselves will be naturally network-aware and transport-layer-independent, a significant achievement.

BRIDE OF LASERSTEIN IN 3-D

As we whispered quite some time ago, start-up Holographix finally got some venture capital to produce their new-generation laser printer/scanner engine. This engine is unique in that a holographic disk replaces the usual series of lenses and rotating polygonal mirrors that scan incoming images. Fewer moving parts mean fewer maintenance worries. Expect to see the fruits of Holographix's labor in a very inexpensive 400 dpi Postscript printer in about two years.

UNBLURRING THE DISTINCTION

The University of Rochester researchers have devised an algorithm that converts blurry photographs into crisp, detailed images. The method involves digitizing the image and calculating the

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

COMMON LISP WITH CLASS

Defclass, that is.
Star Sapphire Common LISP
now has the Common Lisp Object System,
the new standard for object oriented
programming for Common Lisp.

Star Sapphire LISP has other
classy features like extended and virtual
memory support, so you can have
up to eight megabytes of LISP workspace
without breaking your budget.

We outclass the competition.
We have twice as many Common LISP
functions as any similar product
for the PC. In addition,
you get a resident EMACS editor and
hypertext language reference, features that
other PC LISPs forgot.

All this at a price you wouldn't expect for
such a class act: only $99.95.

So go to the head of your class.
Get Star Sapphire LISP. For PCs, PS/2s
and compatibles, running DOS.
Available now from
Sapiens Software Corporation,
P.O. Box 3365, Santa Cruz, CA 95063.
(408) 458-1990, fax (408) 425-0905.

SOFTWARE FOR EVOLVING MINDS

Circle 13 on Reader Service Card

INSTALL.BAT

First impressions last.

INSTALL 3.0 gives your software product a professional introduction.

Installations that rely on batch files look amateurish and have limited error handling. INSTALL 3.0 creates an elegant installation procedure that will increase user's confidence in your software product.

INSTALL 3.0 comes with many samples you can modify and use immediately; no programming usually needed. All available RAM is used for fast file transfers.

- Features very sophisticated error-handling
- Does high performance data compression
- Can safely and reliably modify system files
- Allows selective installation of parts of your program
- Complete source code included
- No royalty charges
- Free technical support

INSTALL 3.0 detects monitors, DOS versions, coprocessors, etc. to create machine-specific installations without asking questions.

INSTALL 3.0 has been used for over four years by some of the biggest (and smallest!) names in the industry, in the U.S. and abroad, to install millions of copies of programs. Make sure your software product makes a good first impression with INSTALL 3.0.

KNOWLEDGE DYNAMICS CORP.
Highway Contract 4, Box 185-H
Canyon Lake, TX 78133-3508
MasterCard/VISA/COD/POs welcome

$249.95 + shipping
Money back guarantee
Sales 800-331-2783 Ext 0196
Internat'l 512-964-3994
24 hr FAX 512-964-3958
24 hr BBS 512-964-3929

Circle 14 on Reader Service Card

BEYOND COMPARISON

Go beyond the clumsy performance and limited features of traditional file comparison utilities . . . DELTA provides a state-of-the-art comparison tool. Invaluable to all DOS users, DELTA is a must for programmers.

- Intelligent Alignment
- Built-in Editor
- Intuitive User-interface
- Directory Comparison

Requires 384K and DOS 2.0 or higher. A hard disk is recommended.

With DELTA you can
see the difference at a glance.
Windowed views stay in synch
Use the built-in editor
while viewing the comparison

To order call
800•542•0938

MasterCard or VISA Accepted
$100
30-DAY MONEY-BACK GUARANTEE

DELTA
THE COMPARISON TOOL

OPEN Network
POWER TOOLS FOR POWER USERS

5 Berkeley Place ▶ Brooklyn, NY 11217 ▶ Voice: 718•398•3838 ▶ BBS: 718•638•2239

Circle 15 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

speed of the object that caused the blur and the degree to which the camera was out of focus. By sorting and filtering the image, a crisp and clear new negative is made. I'm sure this technology will make it into the consumer electronics market once digital camera prices come down from the stratosphere.

CARRY A BIG DISK

Maxtor of San Jose has announced the availability of their Panther 5.25-inch disk drives. Not only can these SCSI-2/ESDI drives store tremendous amounts of data (up to 1.67 gigabytes) in a small space, they have super-fast average access times of 10.5 msec.

Not to be outdone, Conner Peripherals has shown their Summit series half-height 3.5-inch drive that can store up to 510 megabytes and has an average access time of 12 msec. The suggested list price is $1,295. This would be a perfect addition for some high-end laptop machines.

THE RIGHT CHOICE

As a side note, I fail to understand the reasoning behind AT&T's decision to support competitor OSF's user interface in Unix software's future releases. Doesn't AT&T realize they're falling prey to the whole OSF plan of fragmenting the Unix market? OSF will have AT&T spinning their Unix wheels, wasting valuable resources by not working on selected, strategic Unix technology while DEC, IBM, and HP have time to map out their own game plans. Then again, like Xerox, AT&T has never been known for their marketing acumen.

MACH ON MAC

Tenon Intersystems of Santa Barbara has implemented the Carnegie-Mellon operating system, Mach, over the Apple Macintosh's. This is the operating system that runs Steven Jobs's NeXT computer and offers a Berkeley Unix system call interface (but is quite different internally). It doesn't need a memory management unit such as A/UX, and it can run on a small machine like a Macintosh Classic. A two-user license for this product, Mach Ten, costs $595.

XEROX SUES FOR COPYING

Sounds funny, doesn't it? But Xerox sued Metaphor Computer Systems of Mountain View, California, for infringing on the Star and Viewpoint graphical user interfaces that were developed at PARC in the early 1980s. David Liddle, the chairman of Metaphor and a former Xerox PARC employee, was a member of the team that developed the Star user interface technology. IBM was extremely interested in Metaphor's neat graphical interface to relational databases and now owns nine percent of Metaphor.

EYE SEE YOU

Tektronix recently introduced a reasonably high-resolution, 1024 × 1024 pixel silicon-based charged coupled device (CCD). The light sensitivity of this device is more than 70 times that of the human eye. This integrated circuit can also detect more than 4,000 levels of gray scale.

Kodak has a single-chip 4.2-megapixel CCD imager. These devices could someday be used as biological implants (said in my very best Isaac Asimov voice) in order to increase your vision. Some time ago, a neurosurgeon friend told me about experiments that explored the potential of artificially stimulating the brains of certain blind people by using little CCD cameras (which were mounted in a pair of eyeglasses) to generate a coarse matrix of "pixels."

This was enough to enable these people to detect large moving objects (such as cars and buses). The next step would be to increase the resolution of the CCDs and then add some computing power to decipher the images and fire off the "correct" pixels in the brain, yielding a more accurate image.

Optronics Engineering has also developed a solid-state color CCD video camera that is capable of "shutter speeds" equivalent to 1/10,000th of a second. Resolution is 768 × 493 pixels and the device is ⅔ inches across the diagonal.

PLANE OLD PLASMA

America's Planar Systems has shown an interesting plasma display. Their 19-inch monitor eschews the usual orange color for high-contrast paper white. Rumors have it that Planar has 8-bit gray scale waiting in the wings.

Although I believe that active matrix LCD displays will eventually win out over plasma video technology (because it potentially can be much, much cheaper to manufacture), it is still an engineering feat to squeeze that much out of plasma.

Thomson Tubes Electroniques of Boulogne-Billancourt, France, has produced an eight-color plasma display—not an easy thing to produce. Thomson will have plenty of military customers for this one.

ALL THE NeWS THAT'S FIT

With the strong relationship between Sun and Xerox (Xerox licenses SPARC technology from Sun, and Sun licenses user-interface technology from Xerox), I wouldn't bet against super-fast PostScript (actually NeWS) printers from Xerox. Their high-end resolution-programmable printers are already capable of two 8.5 × 11 inch 300–400 dpi pages per second—they're actually very fast bit blasters.

By the way, the NeWSprint software ($495) allows non-PostScript output devices to become compatible with PostScript. NeWSprint comes with the same 57 fonts available with Sun's OpenWindows window system.

Alsys has announced that their font editor, Fontographer, will be ported to the Sun environment with the NeWS side of OpenWindows. It will use the OPENLOOK-compliant TNT (The NeWS Toolkit).

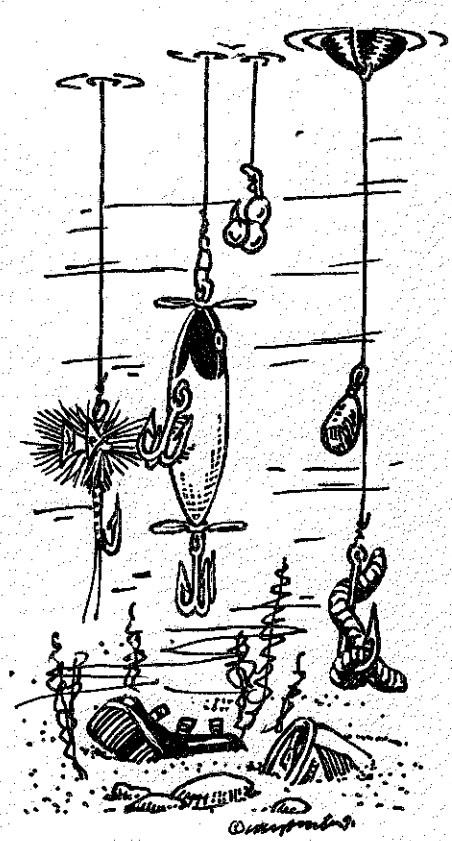
MORE SUG CONFERENCE NOTES

Wayne Rosing, Sun's vice-president of Desktop Systems and Graphics Group, gave an interesting talk on Sun's directions in hardware and software at the Sun User Group Conference. He hinted that in late 1991 Sun will announce a workstation capable of 80–100 MIPS in the standard "pizza-box" desktop configuration. In addition, he speculated that 1000-MIPS Sun machines are only a few years away and that these machines would be uniprocessor versions of the SPARC architecture. A multiprocessor machine running at 1000 MIPS is conceivable, but a single SPARC processor? I'm skeptical of his timetable for this creation.

The real question remains, though—is 1000 MIPS good enough for the user interface? That's a joke, son, I say, just a joke. PJ

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Windows 3 input hooks



Exploring Windows 3 input functions and how to hook them

By Chris Schwartz

This time around, I'll discuss the input system's hook functions and how they can be combined with last issue's MONOSCRN.DLL to create applications that behave much like CodeView for Windows. Instead of jumping right into the code, I'll begin with a look at the event-driven input system in Windows and the hook functions (the SDK documentation spends a mere two pages on these important functions).

Like most GUIs, Windows uses an event-driven input model. Most programmers accept this fact without really digging into the details and benefits of a well-ordered input system. Let's start at the beginning and see how Windows manages the internal input system, how you can hook into it, and how to take advantage of this architecture to create unique system integrations.

When Windows loads, it takes over the keyboard and mouse interrupts and initializes the respective drivers to pass all input interrupts into the USER module at interrupt time. These raw events are placed in the system event queue and sequentially dequeued, decoded, and sent to their appropriate window functions. Before a message is actually returned to an application, it is passed down the hook filter chain. Figure 1 (page 23) displays the message flow.

There are seven hook types (listed in Table 1). Six are system-wide filters, meaning that they can snoop on messages going to other applications and thus must reside in a DLL. The seventh is WH_MSGFILTER, which only sees messages sent to the application that installed it. This can be located in an EXE file. For this issue's article, we're only going to concern ourselves with the WH_GETMESSAGE hook.

TSR WINDOW APPLICATIONS

If you think these hook filters look like the DOS wonderland of TSR programming, you're right—but we don't have to jump on top of hardware interrupts or worry about upsetting Sidekick. These hooks are easier to implement; we are able to trap the **SendMessage** function (WH_CALLWNDPROC), so we're able to monitor application switching and manage the standard input messages.

Just as in TSR programming, there are four things we can do inside a hook filter procedure:

- the infamous **do nothing** choice;
- modify a message before it is dispatched;
- discard the message altogether;
- make a note of the event and allow it to flow through the system normally.

The fourth option is the safest way to respond to a HotEvent (KBD or mouse action) in a hook filter, because we don't meddle with the message stream. It is also the easiest; all we have to do is call **PostMessage** to acknowledge the

TABLE 1: Hook filter types

Name	Scope	Description
WH_CALLWNDPROC	System	Gets hit every time SendMessage is called.
WH_GETMESSAGE	System	Gets hit every time GetMessage returns.
WH_KEYBOARD	System	Gets hit every time a KBD event is pulled from GetMessage or PeekMessage.
WH_JOURNALRECORD	System	Used to record a sequence of events for later playback.
WH_JOURNALPLAYBACK	System	Playback the events captured by WH_JOURNALRECORD.
WH_SYSMSGFILTER	System	Gets hit during the modal loop for MessageBoxes, Dialogs, and Menus.
WH_MSGFILTER	Application	Gets hit during the modal loop for MessageBoxes, Dialogs, and Menus.

Chris Schwartz is president of Softworks, which specializes in Windows and OS/2 software development. Contact Chris at (404) 876-6115.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Develop Powerful Image-based Applications Quickly and Easily

with

Victor Image Processing Library

Powerful Image Processing

VICTOR simplifies development of scientific imaging, quality control, security, and image database software. Victor gives you powerful image processing (brighten, alter contrast, sharpen, outline, linearize, matrix convolution, overlay, resize, flip, mirror, and much more), screen display, file handling, and the ability to hold multiple images in expanded memory.

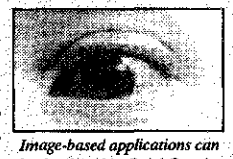


Image-based applications can be developed in QuickC and Turbo C environments.

Gray Scale and Color Images

Victor allows you to handle 8-bit color and gray scale images from any source, as TIFF/PCX/GIF files, and display single or multiple images on EGA/VGA up to 1024x768x256. Your software can have features such as: zoom, resize, display multiple images, pixel editing, image processing, and laser printing. And, to get you up and running, we've included Zip Image Processing software for rapid testing and prototyping of image processing and display functions.

Victor supports Microsoft C, QuickC, and TurboC, includes demonstration and prototyping software, and full documentation. Source code available.

Your application will be able to print halftones at any size.

Victor Library $195

Call (314) 962-7833 to order

VISA/MC/COD

Catenary Systems 470 Belleview St Louis MO 63119 (314) 962-7833

Circle 16 on Reader Service Card

We Have Ways To Increase Your Productivity

TE Developer's Kit

Why reinvent the wheel? TE Developer's Kit provides the most cost effective way of incorporating text editing features into your application. The kit includes the complete source code for the editor and the executable file. Supports Microsoft and Borland 'C' compilers.

Features: multiple files/windows, word-wrap and printing, edits files larger than available memory, reconfigurable keyboard commands and screen colors, undo, column block commands, and search/replace. The kit also includes TER small editor routine. TER is a subset of TE. Your application calls this routine by specifying the window location and size, maximum file size, and an input buffer or file. *IDOS: $169, Windows: $269)*

WinMem: Windows 3 Memory Manager

WinMem provides an unlimited amount of virtual memory even in Windows' Real and Standard modes. The device independence is not enough. WinMem offers you CPU and memory resource independence. It provides an unlimited number of global memory handles, and reduces the overhead per block to 4 bytes (from 20 bytes of Windows). WinMem will trap any attempt to free an illegal memory block. This feature alone can save you a countless number of hours of debugging memory problems. Includes the complete source code. *($139)*

Team: The Programmer's Text Editor

An advanced text editor for a not so advanced price. **Features:** edits unlimited size files, unlimited levels of undo/redo, multiple zoomable windows, regular expression search/replace, editable keystroke macros, 'C' like macro language, task switching, customizable context sensitive help, customizable keyboard commands, menu with accelerator keys, frees entire memory for DOS commands, 132 column screen mode, and much more. *($139)*

Sub Systems, Inc. 1-800-447-6819

Fax: 1-617-438-8901

159 Main Street, #8C, Stoneham, MA 02180, 1-(617)-438-8901

Circle 17 on Reader Service Card

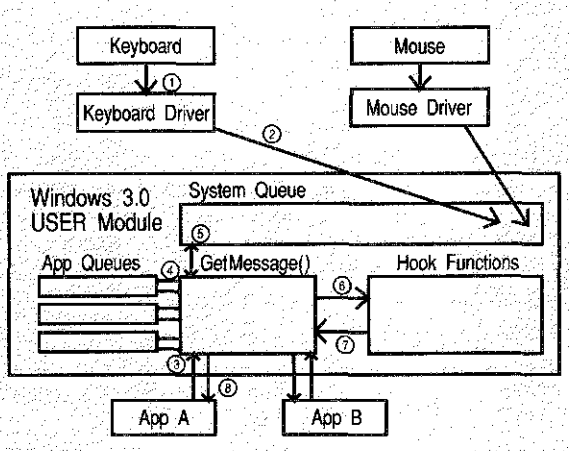
LISTING 1

```
/* MONOICON.C */
#include "windows.h"
#include "monoscrn.h"
#include "gmhook.h"

#define WM_HOTEVENT     (WM_USER + 0x1000)

//function prototypes
int       PASCAL WinMain(HANDLE, HANDLE, LPSTR, int);
          BOOL   InitApplication(HANDLE);
          BOOL   InitInstance(HANDLE, int);
long FAR PASCAL MainWndProc(HWND, unsigned, WORD, LONG);
void      PASCAL ConvertToRowCol(POINT *, POINT *, int, int);

HANDLE  hInst;
BOOL    bOnMonoScreen = FALSE;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance,
                   LPSTR lpCmdLine, int nCmdShow)
{
    MSG msg;

    if (!hPrevInstance) {
        if (!InitApplication(hInstance))
            return (FALSE);
    } else return(FALSE);

    if (!InitInstance(hInstance, nCmdShow))
        return (FALSE);
    while (GetMessage(&msg, NULL, NULL, NULL)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    whRemoveGMHook();                   //REMOVE the GM Hook !!!
    return (msg.wParam);
}

BOOL InitApplication(HANDLE hInstance)
{
    WNDCLASS  wc;

    wc.style         = CS_DBLCLKS;
    wc.lpfnWndProc   = MainWndProc;
    wc.cbClsExtra    = 0;
    wc.cbWndExtra    = 0;
    wc.hInstance     = hInstance;
    wc.hIcon         = LoadIcon(NULL, IDI_APPLICATION);
    wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground = GetStockObject(WHITE_BRUSH);
    wc.lpszMenuName  = NULL;
    wc.lpszClassName = "MonoIconClass";
    return(RegisterClass(&wc));
}

BOOL InitInstance(HANDLE hInstance, int nCmdShow)
{
    HWND    hwnd;

    hwnd = CreateWindow("MonoIconClass", "Gateway to Mono Monitor",
           WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT,
           CW_USEDEFAULT, CW_USEDEFAULT, NULL, NULL, hInstance, NULL);
    if (!hwnd)
       return (FALSE);
    VioInit();
    whInstallGMHook(hwnd, WM_HOTEVENT);         //INSTALL the GM Hook !!!
    whEnableGMHook(TRUE);
    ShowWindow(hwnd, SW_SHOWMINNOACTIVE);       //make window ICON
    UpdateWindow(hwnd);
    return(TRUE);
}

long FAR PASCAL MainWndProc(HWND hwnd, unsigned msg, WORD wParam,
    LONG lParam)
{
    POINT ptTemp;
    POINT ptMono;

    switch (msg) {
        case WM_QUERYOPEN:
            return(FALSE);                      //stay as an ICON
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
```

continued

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

FIGURE 1: Keyboard and mouse message flow



Keystroke Event Processing

① Key Press
② Driver Updates System Queue
③ App Calls GetMessage()
④ GM() checks App Queues
⑤ GM() checks System Queue
⑥ GM hands MSG to Hook Functions
⑦ Hook Functions return modifications
⑧ MSG is delivered to App

trigger in our application queue. A subsequent call (probably the next) to **GetMessage** will dequeue our internal acknowledge message; then our application can call **SetWindowActive**.

It's best to continue to think of hook filter functions as TSRs, even though they are not. As system-wide filters, they will get hit often and so should be small and quick. Other hook filter strategies are to queue the messages internally—as the *spy* utility does—and run a timer to the client window to dequeue the messages. This teaches you how Windows manages itself.

Another option is to put the function into the actual hook and not worry about affecting system performance. This is only an option for building custom debugging tools. I've used this approach in building an internal data structure debugging tool that can bind itself to applications during development. This is almost a necessity in Windows, because it often fails to protect wild writes. If we wanted to get fancy

```
        case WM_MOUSEMOVE:
            if (bOnMonoScreen) {
                ptTemp.x = LOWORD(lParam);
                ptTemp.y = HIWORD(lParam);
                ClientToScreen(hwnd, &ptTemp);
                ConvertToRowCol(&ptTemp, &ptMono, 25, 80);
                VioMovePointer(ptMono.y, ptMono.x);
            }
            break;
        case WM_RBUTTONDBLCLK:
            if (bOnMonoScreen) {
                whEnableGMHook(TRUE);
                ReleaseCapture();
                ShowCursor(TRUE);
                VioShowPointer(FALSE);
                bOnMonoScreen = FALSE;
            }
            break;
        case WM_HOTEVENT:
            if (hwnd != GetActiveWindow())
                SetActiveWindow(hwnd);
            whEnableGMHook(FALSE);
            SetCapture(hwnd);
            ShowCursor(FALSE);
            VioShowPointer(TRUE);
            bOnMonoScreen = TRUE;
            break;
        default:
            return (DefWindowProc(hwnd, msg, wParam, lParam));
    }
    return(NULL);
}

void PASCAL ConvertToRowCol(POINT *ptScreen,POINT *ptMono,int iRow,int iCol)
{
    static BOOL bInit = 0;
    static int  cxScreen = 0;
    static int  cyScreen = 0;
           int  iXConvFactor = 0;
           int  iYConvFactor = 0;
    if (!bInit) {
        cxScreen = GetSystemMetrics(SM_CXSCREEN);
        cyScreen = GetSystemMetrics(SM_CYSCREEN);
    }
    iXConvFactor = cxScreen / iCol;
    iYConvFactor = cyScreen / iRow;

    (*ptMono).x = ptScreen->x / iXConvFactor;
    (*ptMono).y = ptScreen->y / iYConvFactor;

    // fixup the math because were using INT instead of real numbers.
    if (ptMono->x >= iCol)
        ptMono->x = iCol-1;
    if (ptMono->y >= iRow)
        ptMono->y = iRow-1;
}
```

LISTING 2

```
/* GMHOOK.C */
#include "windows.h"
#include "gmhook.h"

FARPROC     lpfnOldGMHook  = 0;
HWND        hwndCallBackGM = 0;
BOOL        bGMHookActive  = FALSE;
unsigned    usMsgGM        = 0;
WORD        _acrtused      = 0;

DWORD  FAR PASCAL whGetMessageHook(int iCode, WORD wParm, LPMSG lParm)
{
    if (iCode < 0) {
        return(DefHookProc(iCode, wParm, (LONG)lParm, &lpfnOldGMHook));
    }
    if ((iCode == HC_ACTION) && bGMHookActive) {
        if (lParm) {
            if (lParm->message == WM_NCRBUTTONDBLCLK) {
                PostMessage(hwndCallBackGM, usMsgGM, 0, 0L);
            }
        }
    }
    return(DefHookProc(iCode, wParm, (LONG)lParm, &lpfnOldGMHook));
}

VOID FAR PASCAL whInstallGMHook(HWND hwnd, unsigned msg)
{
```

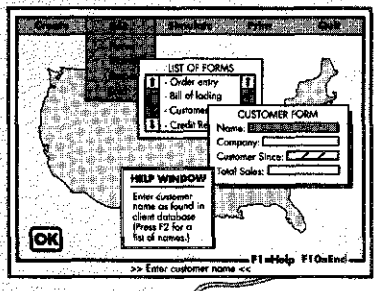
continued

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

POWERFUL SCREENS FOR POWERFUL PROGRAMS

HI-SCREEN XL is a powerful screen management system with a high-level modular approach that provides for superior programming **control, flexibility and speed.**

- The built-in **screen editor** is used to create screens/windows/menus along with all field descriptions. Any **Hercules, CGA, EGA** or **VGA** graphics can also be captured. Screens are then saved in a compressed file format outside your program.
- **High-level calls** from your program give you total control over the user interface. Combine text and graphics, dialog boxes, context-sensitive help, elaborate data entry & menu management, keyboard & mouse control, etc.



- Supports all versions of **C, Basic, Cobol, Fortran, Pascal, dBase, Assembly,** etc.
- Use linkable (with C and QuickBasic) or removable TSR screen driver (30-40K).
- 2 compatible versions:

HI-SCREEN XL.....$149
HI-SCREEN XL *Professional Series*
(with mouse & graphics support):.....$325

30 day money-back guarantee • No royalties

Call today: **1-800-338-2852**
in CA: 415-896-0708 • Fax: 415-896-0709

SOFTWAY, Inc.
185 Berry Street, Suite 5411,
San Francisco, CA 94107

CALL FOR DEMO

Circle 18 on Reader Service Card

we could create a linked data structure of HotEvents, activate specific applications, and possibly execute a macro command—which is similar to what is done by the Recorder Accessory or the Bridge Toolkit, both by SoftBridge.

AN ICON GATEWAY

The example application (Listing 1) is fairly simple. It creates an icon window that serves as a gateway to the monochrome monitor. The WM_QUERYOPEN message in **MainWindowProc** is interesting. By returning **false** all the time, the application will always display as an icon. Notice that instead of passing **nCmdShow** to **ShowWindow,** I used SW_SHOWMINNOACTIVE.

On start-up the application installs a **GetMessage** hook (Listing 2) that posts a message to signal the detection of a WM_NCRBUTTONDBLCLK. This means we can jump to the monochrome monitor from any application by double-clicking the right button on the title bar. Once on the screen, we wait for another right-button double click and move the pointer back to the Windows screen.

Listing 3 contains the new functions to display a pointer on the monochrome monitor. The code that processes the WM_HOTEVENT message is straightforward. First, we activate our icon. Because we are in an icon state, we also activate the keystroke characters (the KBD messages come as WM_SYSKEYUP, etc.). Second, we disable the hook. Third, we grab the mouse capture and turn the Windows mouse pointer off. As the last step we turn on the monochrome monitor pointer, which is then updated by WM_MOUSEMOVE. A WM_BUTTONDBLCLK is our signal to reverse everything, which brings us back to normal.

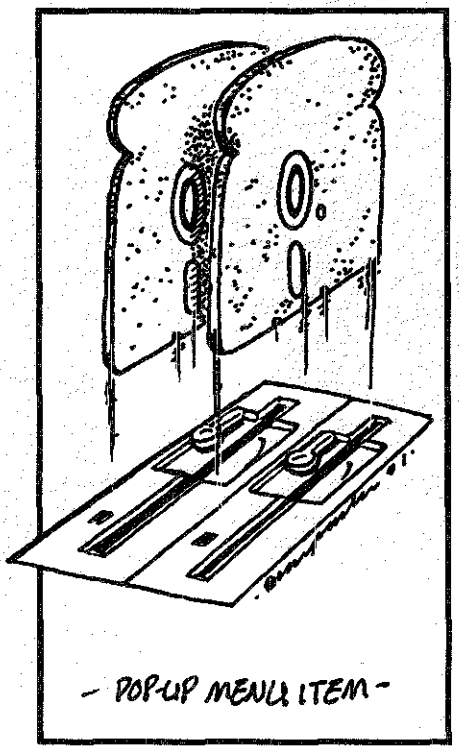
Almost everyone familiar with Windows understands the value of DDE and DLLs in extending shrinkwrap software. I call these "back-end" application extensions because they depend on the application to which they are tied to trigger the action. I consider hook filter functions to be the mechanism for "front-end" extensions, because they get hit before the application gets a chance to respond to the message. Combining these front- and back-end mechanisms is the key to creating highly integrated application workstations with generic shrinkwrap software and a dab of custom glue code.

NOT-SO-BLIND FAITH

In the listing that acompanied my *PJ* 9.1 article, the code used the DPMI segment to the selector function to gain access to the monochrome memory buffer because the pseudo-selectors were causing problems. This is not the case now, but because I no longer have the code that produced the problems, I don't have an explanation. Both techniques work equally well, because they return the same value. I only mention this now because I don't want to create the impression that those values don't work—they do.

For an additional bonus, try running the EXEHDR utility on KERNEL.EXE. You will see that there are three more pseudo-selectors than the ones listed in the SDK. Running EXEHDR -V on CVW.EXE (-V shows the imports) results in *__B000h*, which is a reference to KERNEL.181.

Based on past experience, perhaps I don't put as much faith in Microsoft's documentation as I should. Anyway, the moral of this story is that, in the Windows environment as in the big bad world, there is more than one way to skin a cat. PJ



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Tear Your C Development Time in Half.

We understand the pressure you're under, trying to create, test, and debug complex C programs without Instant-C™, the professional development environment. With all your searching for errors and recompiling, you barely have time to test code, never mind read magazines.

Instant-C can make your time twice as productive. Here's how:

Industrial-strength debugging. Instant-C finds bugs other compilers miss, automatically detecting over 500 errors, including:

- *Array references out of bounds*
- *Stack overflow*
- *Use of uninitialized pointers*
- *Overwriting a return address*
- *Read or write through null pointers*
- *Overwriting your program or DOS*

Ultra-fast, incremental compiling. When you make a change to a program, Instant-C recompiles just the change. In fact, Instant-C can rebuild a 100,000 line program in less time than it takes to read our headline. And in recent tests, Instant-C was 24 times faster than Microsoft Programmer's WorkBench.*

Microsoft® C compatibility. Instant-C links with Microsoft C libraries to form an enhanced development environment. After developing, testing, and debugging with Instant-C, you can use Microsoft C to make the final version of your program.

Unlimited program size. Instant-C uses Rational Systems' DOS/16M technology, the software that enabled Lotus to break the 640K barrier for 1-2-3® Release 3. Because it uses extended memory, no program is too big, and your work isn't slowed down by swapping to disk.

That's not the half of it. Instant-C is a complete development environment, including source-level debugging, dynamic cross referencing, lint-like static check, support for third-party libraries, and interactive testing of single functions or partial programs.

The Instant-C Environment

- The industry's fastest edit-compile-link-debug cycle
- Configurable editor and source formatter
- Incremental compiler and linker
- *Direct access to 16MB of memory*
- Extensive static and run-time error checking
- Multiple virtual or real screens
- Source-level debugging
- Support of commercial libraries
- Conditional breakpoints in C
- Compatibility with assembly language object modules
- Immediate evaluation of C expressions, including macros
- Compatibility with Microsoft C compiler 5.1 and 6.0
- Interactive execution of functions
- Allows use of Microsoft run-time libraries
- Dynamic cross-referencing and browsing
- Full graphics screen support

Get the whole story. To find out more about how Instant-C can double your productivity, give us a call. We'll give you all the details in a flash. After all, we know how valuable your time is.

Rational Systems, Inc.

Rational Systems, Inc.
220 N. Main St.
Natick, MA 01760
Toll-free: (800) 666-2585
Outside U.S.: (508) 653-6006
Fax: (508) 655-2753

* Tests involved changing one line of code in a 11,500-line program, running on a 20MHz 80386 system with 8MB memory. Instant-C requires an 80286/386/486 and at least 1M of memory. Instant-C is a trademark of Rational Systems, Inc. Microsoft is a registered trademark of Microsoft Corp. 1-2-3 is a registered trademark of Lotus Development Corp.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Circle 19 on Reader Service Card

SPOTLIGHT ON SUCCESS:

Our goal was to create a comprehensive mine planning system for our underground mining operations. Using SilverScreen's C compiler we have developed a mine planning system that uses SilverScreen as its CAD platform. Our system is two years ahead of others like it.

Peter S. Campbell
Mines Computer Systems
Senior Project Engineer
Falconbridge, Limited

CAD ENGINE

SilverScreen Licensing Program

SilverScreen is a 3D CAD and Solid Modeling software system that contains a resident C Language compiler. The C compiler was designed by Schroff Development for use by end users and software developers to facilitate the development of custom applications requiring 3D and solid modeling.

SilverScreen is the most powerful and versatile 3D CAD/Solid Modeling system available containing such features as: 2D and 3D Boolean operations; rendering; shading; mass properties; text editor; associative dimensioning (ANSI Standard); object-oriented database; schema-based attributes; and camera walk. SilverScreen runs on DOS and UNIX (Silicon Graphics) and supports a wide range of mice, video boards, printers, plotters and digitizers. File transfer is handled through DXF, IGES, HPGL, PostScript and others. Silverscreen features direct export to SmartCAM and Stereolithography (3D Systems).

SilverScreen's resident C compiler implements a library of more than 300 functions plus a large number of functions dedicated to accessing the SilverScreen database (from which parts lists and bill of materials can be generated) and to controlling the SilverScreen environment. Custom menus and icons also may be created. SilverScreen also adds the powerful ability to embed the SilverScreen script commands in a program for direct execution by SilverScreen.

For SilverScreen information contact:

Schroff Development Corporation
P.O. Box 1334
Mission, KS 66222
(913) 262-2664
Dealer Inquiries invited
Evaluation packages available

SilverScreen®

3D CAD/Solid Modeling Software

```
    lpfnOldGMHook = SetWindowsHook(WH_GETMESSAGE, (FARPROC)whGetMessage-
Hook);
    hwndCallBackGM = hwnd;
    usMsgGM        = msg;
}
VOID FAR PASCAL whRemoveGMHook(VOID)
{
    UnhookWindowsHook(WH_GETMESSAGE,
                      (FARPROC)whGetMessageHook);
}
BOOL FAR PASCAL whEnableGMHook(BOOL bEnable)
{
    return(bGMHookActive = bEnable);
}
int FAR PASCAL LibMain(HANDLE hModule,  WORD wDataSeg,
                       WORD cbHeapSize, LPSTR lpszCmdLine)
{
    return(1);
}
int FAR PASCAL WEP(int bSystemExit)
{
    return(1);
}
```

LISTING 3

```
; additional functions needed in MONOSCRN.ASM
; also add these to the EXPORTS in MONOSCRN.DEF
                PUBLIC  VioMovePointer
                PUBLIC  VioShowPointer

;       FAR PASCAL Calling Convention
;       VioMovePointer(wRow, wCol)
;------------- Formal Parameters ----------
wRow            equ     [bp+06]
wCol            equ     [bp+06]

VioMovePointer  PROC    FAR
                push    bp
                mov     bp, sp
                push    ds
                mov     ax, DGROUP
                mov     ds, ax                  ;fixup dll's data segment
                push    es
                push    di
                push    si

                cmp     lShowPtrLevel, 0
                jle     VMPJustUpdate

                ;need to restore the old attr and update the new location
                mov     ax, iRow
                mov     dx, iCol
                call    InternalHidePointer

                mov     ax, wRow
                mov     dx, wCol
                xor     si, si
                call    InternalShowPointer
VMPJustUpdate:  mov     ax, wRow                ;update internal values
                mov     dx, wCol
                mov     iRow, ax
                mov     iCol, dx
VioMovePointerExit:
                pop     si
                pop     di
                pop     es
                pop     ds
                pop     bp
                ret     4
VioMovePointer  ENDP

;       AX = row        DX = col
InternalShowPointer     PROC    NEAR
                call    ScreenOffset
                mov     es, dx
                inc     ax
                mov     di, ax
                mov     ax, WORD PTR es:[di]
                mov     iOldAttr, ax
                rol     al, 4
                stosb
                ret
```

continued

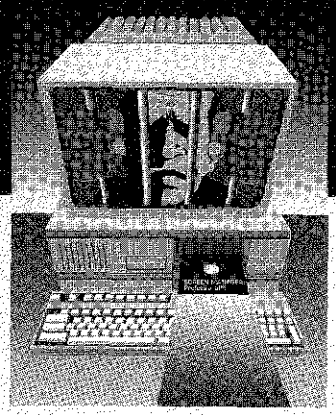
Circle 20 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
InternalShowPointer     ENDP

;       AX = row        DX = col
InternalHidePointer     PROC    NEAR
                call    ScreenOffset
                mov     es, dx
                inc     ax                      ;move over to attr byte
                mov     di, ax
                mov     ax, iOldAttr
                stosb
                ret
InternalHidePointer     ENDP

;       FAR PASCAL Calling Convention
;       VioShowPointer(bTrueFalse)
;------------- Formal Parameters -------------
bTrueFalse      equ     [bp+06]

VioShowPointer  PROC    FAR
                push    bp
                mov     bp, sp
                push    ds
                mov     ax, DGROUP
                mov     ds, ax                  ;fixup dll's data segment
                push    es
                push    di
                push    si

                mov     ax, bTrueFalse
                or      ax, ax
                je      VSPDecLevel

                inc     iShowPtrLevel
                cmp     iShowPtrLevel, 0
                jbe     VioShowPointerExit

                mov     ax, iRow
                mov     dx, iCol
                call    InternalShowPointer
                jmp     SHORT VioShowPointerExit
VSPDecLevel:    dec     iShowPtrLevel
                cmp     iShowPtrLevel, 0
                jne     VioShowPointerExit

                mov     ax, iRow
                mov     dx, iCol
                call    InternalHidePointer
VioShowPointerExit:
                pop     si
                pop     di
                pop     es
                pop     ds
                pop     bp
                ret     2
VioShowPointer  ENDP
```

LISTING 4

```
; gmhook.def
LIBRARY gmhook
EXETYPE WINDOWS
CODE    PRELOAD FIXED
DATA    PRELOAD FIXED SINGLE
EXPORTS
    WEP                     @1  RESIDENTNAME
    mhGetMessageHook        @2
```

Free Yourself!

With Screen Manager Professional™

Concentrate on the substance of your programs with the advanced interface design library for C programmers.

Screen Manager Professional gives your applications the finest bullet proof windowing interface available. All the functions have been carefully designed and tested with the serious programmer in mind The functions are easy to use and remember. The elegant and powerful functions provide the fastest possible response and smallest possible size. Free yourself to concentrate on the substance of your program. Over 200 functions provide all the windowing tools needed to develop serious applications whether you are a professional or a hobbyist.

SMP gives your applications:

- Windowing
- Menus
- Mouse Support
- Field oriented data entry
- Interactive field building utility
- Keyboard Support Functions
- Context Sensitive Help with Hypertext
- Low RAM Overhead / High Speed
- User attachable validation functions

Linkable with: Microsoft C - Zortech C, C++ - Quick C
Turbo C, C++ - Watcom C

Only $349.95 **Source code available**

SCREEN MANAGER Professional™

MAGEE™

Call today for a free demonstration diskette
1-800-662-4330

P.O. Box 1587
Norcross, GA 30091 USA
404-446-6611
BBS 404-446-6650

Circle 21 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C compilers for Windows

Comparing C compilers that support Microsoft Windows development

By Paul L. Yao

This article looks at Windows-compatible C compilers. Two C++ compilers are included because of their ability to work with native C code. Some other language compilers that are capable of creating Windows-compatible code are not covered here—for example, Modula-2 compilers from JPI, Stony Brook, and Logitech.

If you're using Microsoft's C compiler (version 4.0 or later), creating Windows-compatible code is no problem. After all, Microsoft's C compiler and Windows grew up together. In fact, parts of Windows were created using Microsoft's compiler.

If you've been using any other vendor's compiler, you may still be able to use it to develop Windows applications. As shown in Table 1, quite a few of the commercially available compilers can create Windows code. Notable because of its absence from this list is the Lattice compiler. If you have a Lattice C compiler and you wish to create Windows programs, you'll have to acquire one of the other compilers on the list; I telephoned Lattice and was told that they have no plans to create a Windows-compatible compiler.

Paul Yao is president of International Systems Design, a firm specializing in Microsoft Windows consulting, software development, and training. He has co-authored two Windows programming books, including Peter Norton's Windows 3.0 Power Programming Techniques *from Bantam. Mr. Yao is technical chairman for this summer's Boston University Windows Developers' Conference.*

MAKE MINE MIN

My test for Windows compatibility is simple: a Windows-compatible compiler must be able to create a working version of MIN, a minimum Windows program. It served as the starting point for every program in *Peter Norton's Windows 3.0 Power Programming Techniques*. The six files that make up the minimum Windows program, MIN.EXE, are listed in Table 2. The first four are text files and are shown in their entirety (see Listings 1–4). The last two files are graphic objects: *min.ico* contains an icon that is displayed when MIN's main window is minimized; *min.cur* contains a cursor—a graphic image used as a mouse pointer.

The files in Table 2 were created and tested using Microsoft's C compiler, version 6.0. Some of the tested compilers required changes to MIN.C in order to create a Windows program. In other cases, no changes were required. In my opinion the fewer the changes the better, because you can use the relatively large body of existing sample source code for your own development projects. In all cases, however, the changes are straightforward and shouldn't take too much time to put in place once you know what they are.

If you're an experienced Windows programmer, you probably think my test is too simple. You're right: a complete test would check whether the compilers can create dynamic link libraries (DLLs), and I didn't test for this.

The primary reason I chose not to is that most Windows development projects don't require DLLs. After all, they are required only for larger projects or projects that involve the creation of custom device drivers. It's my experience that most programmers don't create DLLs. If you find yourself working on a project that requires DLLs, you'll want to run some tests to make sure your compiler can meet the challenge.

WINDOWS PROGRAMMING

What is necessary to create a Windows program? The list in Table 3 describes the required tools and the capabilities that each tool must have. The "SDK" column identifies tools that Microsoft ships with its Windows 3.0 Software Development Kit (SDK). With most compilers, you'll need to purchase an SDK to get a complete Windows development environment. The exception is Borland's Turbo C++ compiler, which includes most of the SDK tools or some reasonable substitutes for them.

The column marked "SUPP" identifies the tools Microsoft provides as

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

part of the Windows SDK Supplemental Compiler Utilities Disk. Microsoft makes these tools available to help programmers who are using third-party compilers. The disk is available free of charge to developers. To obtain one, you can call the following toll-free number from the United States and Canada: (800) 426-9400. Developers in other countries should contact their local Microsoft subsidiary for information.

The diskette contains five utilities: LINK is the Microsoft linker for creating Windows programs; LIB is the library management utility; IMPLIB, an import library utility, helps in the creation of dynamic link libraries; EXEHDR, the executable header reader, is used to read a Windows executable file header; and MAKE is used to automate building programs.

CHOOSING A COMPILER

A language compiler must support the Pascal calling convention, support near and far keywords, and create the Windows prolog and epilog. I'll discuss each in turn.

The Pascal calling convention defines the order in which parameters are put on the stack (from left to right) as well as defines stack cleanup (the callee cleans up the stack). A limitation of this compared to the standard C calling convention (as defined by the Microsoft C compiler) is that it requires a fixed number of parameters. An advantage, however, is that it results in smaller and faster code. In fact, when Windows itself was converted from the C calling convention to the Pascal calling convention, the size of the system code actually shrank by 9%.

The near and far keywords define whether a memory reference is a 32-bit **inter**-segment reference (**far**) or a 16-bit **intra**-segment reference (**near**). These keywords are required because all of a Windows program's static data must be accessed as **near** pointers, and pointers that are passed to Windows' library routines must be **far**. In addition, a performance advantage can be gained by tuning a program to minimize **far** references.

The Windows prolog and epilog are additions to those of the standard C. Table 4 shows a comparison between a standard prolog/epilog and a Windows prolog and epilog. The code is shown as it comes out of the compiler. (As you'll find out in a moment, however, the prolog is actually patched by the Windows loader to allow it to play some tricks.)

The Windows prolog contains six new instructions beyond those contained in the standard prolog. These instructions provide support for three separate mechanisms: (1) to set up DS for the called function, (2) to set BP far-call flag for stack walking, and (3) to save the DS of the calling function.

You might wonder why the first two Windows prolog instructions put DS into AX, because the eighth instruction copies the contents of AX back into DS. In fact, the code produced by the compiler is only half of the story. The Windows loader patches **exported functions** so that the prolog appears as follows:

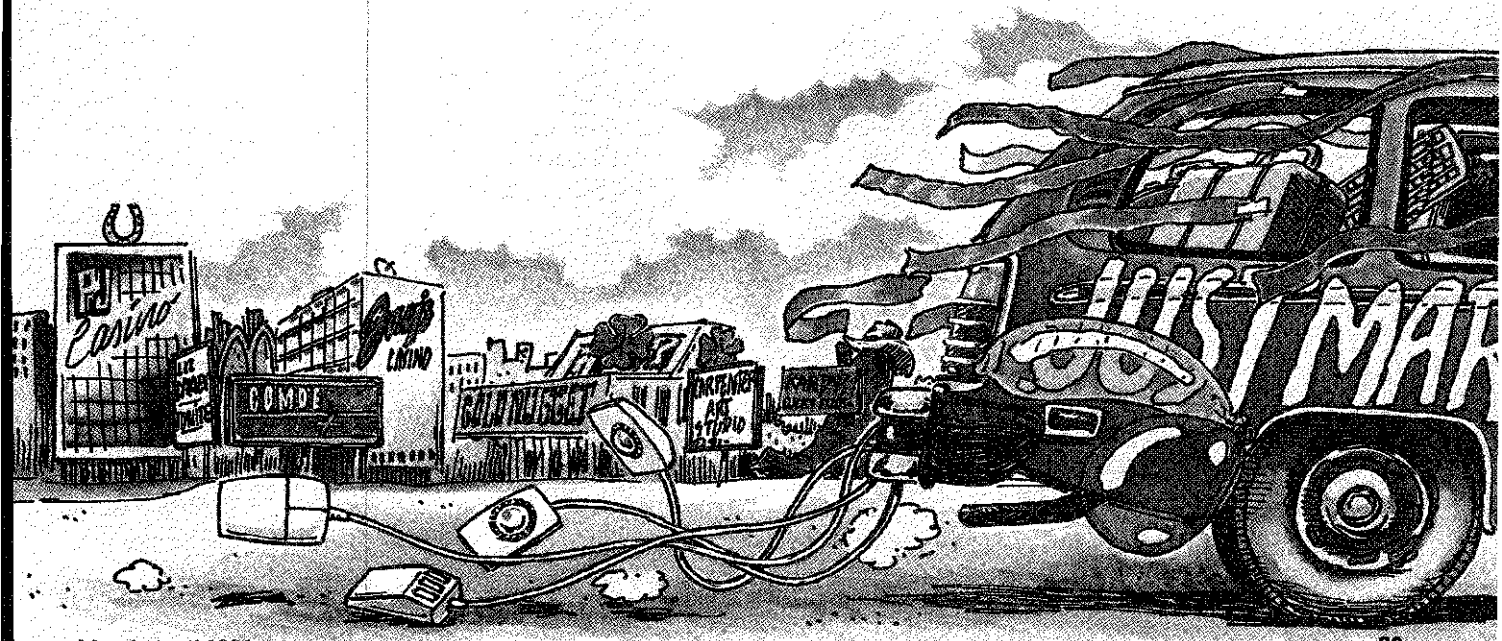
```
nop
nop
nop
inc  bp          ; far-call flag
push bp
mov  bp, sp
push ds          ; save caller's DS
mov  ds, ax      ; install our DS
```

EXPORTED FUNCTIONS

An exported function is one that is declared in the EXPORTS section of a program's module definition (DEF) file. Such functions serve as gateways through which functions in a library or an application can be called.

When an exported function is called, the AX register must contain the correct DS value. Steve Wood, architect of Windows' dynamic linking, describes the process the most clearly: "Every time a module boundary is crossed, DS must be fixed up for the called code." In Windows terminology, a module is either a dynamic link library or an application program.

There are two basic types of exported call-back functions in Windows: window procedures and everything else. When a window procedure is called, the AX fixup is performed by Windows' message-delivery mechanism. In all other cases (dialog box procedures, enumeration procedures, and other call-back procedures), a Windows program



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TABLE 1: Summary of tests for Windows-compatible C compilers

Test	Borland C++ 2.0	JPI C 1.05b	MetaWare C 1.61	Watcom C 8.0	Zortech C++ 2.1	Microsoft C 6.0
Windows compatible	Y	Y	Y	Y	Y	Y
Sample code provided	Y	N	N	Y	Y	N
Compile MIN unchanged	Y	N	N	Y	Y	Y
Size of MIN.EXE (bytes)	6144	6656	4272	7568	6040	4240
Code segment size (bytes)	720	992	1840	592	1840	1808
Uses WINDOWS.H as is	Y	N	N	Y	N	Y
Has Windows libraries	Y	Y	N	Y	N	N
Windows-compatible linker	Y	Y	N	Y	N	Y
Windows-compatible debugger	Y	N	N	N	N	Y

TABLE 2: The six MIN.EXE files

File	Description
min.c	C source code
min.def	Module definition file, used by linker
min.mak	Make file
min.rc	Resource file for user-interface data objects
min.ico	Icon file, a resource
min.cur	Cursor file, another resource

makes explicit calls to the **MakeProcInstance** routine. This creates an "instance thunk," which is a piece of code that looks like this:

```
mov ax,ds_value
jmp  exported_function
```

The second mechanism contained in the prolog is the far-call flag for stack walking. In real-mode Windows, the stack is walked when code segments are moved and return addresses are patched. This allows any code segment to be moved in the system without fear of destroying the return path. It even allows you to discard code segments on the return path, purging them from memory. The **inc bp** instruction lets the stack walker know when a **far** call is being performed vs. a **near** call. Because the stack is always word-aligned, BP is always an even number. This mechanism uses the low-order bit of BP to flag **far** calls so that the stack-walker can distinguish a **far** call from a **near** one. After all, only far-call return addresses need to be patched when segments are moved.

The third mechanism at work in the Windows prolog is the saving of a caller's DS value. This is useful for two reasons. The most obvious is that it preserves a caller's DS across the module boundaries. The less obvious reason is that it allows the stack-walker to patch the DS value when it has to move a program's data segment.

The second and third mechanisms support Windows' stack-patching operation. This allows Windows to overcome some of the onerous restriction of real-mode operation. Stack patching is not performed under protected-mode operation, but the few additional instructions it requires represent a relatively small additional overhead for the considerable privilege of running in all of Windows' operating modes.

A LINKER FOR WINDOWS

When the Windows development team started working on Windows in 1983, Microsoft was just getting ready to ship version 2.0 of DOS—yet DOS 2.0's newly introduced EXE file format couldn't handle the demands of Windows. Specifically, the Windows loader needed a lot of the relocation information that the DOS 2.0 linker discards. A new EXE file format was created for this reason. It was named for the protected-mode operating system that was starting to take shape in 1984: the DOS 4.0 file format. Today, we know that operating system as OS/2. But for several years, the Windows linker was called LINK4, because it was able to create DOS 4.0 (OS/2) executable files.

Regular DOS linkers cannot be used to create Windows programs. You must either have version 5.1 of Microsoft's linker or one that supports this new EXE file format. Four of the six compilers tested (Borland, JPI, Watcom, and Microsoft) come equipped with a Windows-compatible linker.

Although C language programmers are used to dealing with many small **include** files, Windows programmers use a single, large (110K) **include** file. This contains the function definitions, symbolic constants, and data structure definitions needed for Windows programming.

Some of the compilers I tested required minor modifications to this file. In a few cases, no changes were necessary. The Borland compiler, for example, comes equipped with its own tailored version of this file.

The sheer size of this file can cause problems for compilers. In particular, it caused Microsoft's own Quick-C compiler (version 2.5) to fail my tests. It also caused problems for the MetaWare and Zortech compilers.

This problem is serious enough for several compiler vendors to provide large heap-space versions of their products. The Zortech compiler was able to pass my tests only because of a version that used the DOS Protected Mode Interface (DPMI) to get access to more memory, and therefore to a larger symbol space.

STATIC LINK LIBRARIES

There are two pieces that you must have to link a Windows program. One is a file that contains the relocation records needed to link to Windows' dynamic link libraries. In the Windows SDK, this file is *libw.lib*. In general, calling a function in a dynamic link library is the same as calling a function in a static link library. However, at link time a static library function is bound into the executable file of its caller; it becomes owned by the calling program so that it cannot be shared with other programs in the system.

A dynamic link library function, on the other hand, is not bound into the executable file of its caller. Instead, a tiny stub—known as a *dynamic relocation record*—is bound into the code segment of the caller. This stub provides two pieces of information: the name of the dynamic link library and the name of the entry point.

The name might be a function name—such as **TextOut** or **CreateWindow**—or it might be an *ordinal entry point*: a number that identifies the function in the library. In this way, dynamic link

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C CODE FOR THE PC

source code, of course

	GUIDO (Graphical User Interface with Dialog Objects, multiple hot regions per window, data entry functions, window minimization to icon)	$400
	M++ (fast C++ math classes; indexing, matrices & vectors, numerics, memory handling, I/O)	$395
Windows!	C++/Views (C++ interface to MS Windows 3.0; over 60 classes; free broswer (no source); no royalties; Zortech C++ only)	$395
NEW!	MBALLOC (best-fit far heap manager, replaces standard routines, Microsoft performance on Turbo C)	$340
	GraphiC 5.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$325
	QuickGeometry Library (large collection of math, graphics, display & DXF subroutines for CAD/CAM/CAE/CNC; math coprocessor required)	$320
	ZIP Image Processing Library (brightness, contrast, merge images, TIFF/GIF/PCX/bin, printer & HRT 512 digitizer support)	$290
	Victor Image Processing Library (EGA/VGA; TIFF/PCX/GIF/bin; multiple images; many filters; frame grabber & live video)	$250
NEW!	MemCheck Memory Debugger (excellent library to detect buffer overwrites, unfreed pointers, etc., by source file and line)	$250
	TurboTEX (Release 3.0; HP, PS, dot drivers; CM fonts; LaTEX; MetaFont)	$250
	db_File & db_Retrieve Bundle by Raima (B-tree and network database with SQL query and report writer; multi-user $475)	$245
	Install 3.05 (automatic installation program; user-selected partial installation; CRC checking)	$195
	Essential resident C (TSRify C programs, DOS shared libraries)	$190
NEW!	Report Right (report writer and graph generator for Windows 3.0)	$170
	Rogue Wave tools.h++ Class Library (collections, B-trees, file space management, time& date, persistent objects; complete user's guide)	$165
	Rogue Wave math.h++ Class Library (C++ overloadings for standard operators, matrix inversion & FFT; complete user's guide)	$165
	PCX Effects (fades, wipes, animation, music)	$165
	Turbo G Graphics Library (all popular adapters, hidden line removal)	$165
NEW!	DXF Subroutine Libraries (toolkit for making AutoCAD DXF drawing files; includes manual & sample programs)	$155
	WKS Library Version 2.01 (C program interface to Lotus 1-2-3, dBase, Supercalc 4, Quatro, & Clipper)	$155
	Hold Everything (spawn new programs; swap parent to EMS or disk; handles video, interrupts, & environment; returns error level)	$150
NEW!	EDITWIN (C program development editor for Microsoft Windows;)	$150
Windows!	RED (program development editor for Microsoft Windows; keystroke recording & playback; unlimited undo)	$150
	C Generator (generates C code to read & write file records defined with C structure syntax)	$150
	Vmem/C (virtual memory manager; least-recently used pager; dynamic expansion of swap file)	$140
	CBTree (B+tree ISAM driver, multiple variable-length keys)	$135
	C Communications Toolkit by Magna Carta (multi-port & co-processor board support, FAX, interrupt driven, emulations, xfer protocols)	$125
	C++ Object Library (virtual windows, I/O, lists, file free space management, keyed ISAM file I/O)	$125
	Envoy Communications Package (includes editor, script language, redefinable keys, help screens, menu-driven interface; Turbo C)	$120
Windows!	JTW C++ Class Library for Windows 3.0 (over 100 classes, requires Win3.0 SDK & Glockenspeil or Zortech C++)	$120
	Kier CountryLib (date, time, measurement, numeric, and simple language conversion; helps build country-independent software)	$120
	SilverWare "C" EMM Library (60 interface functions to EMM Version 4.0)	$120
	TE Editor Developer's Kit (full screen editor, undo command, multiple windows)	$115
	Network Toolbox/N (C functions for Novell Netware 2.1x, sample programs included)	$110
	B-Strings (dynamic string handling; cut, copy, paste, search, user input, etc.; non-fragmenting memory management)	$105
Updated!	PC/IP (CMU/MIT TCP/IP for PCs; Ethernet, Appletalk & NETBIOS drivers, RVD, Clarkson drivers, Stan's NFS clone, Bdale mailer)	$100
	Disassembler (8086/80286/80386/80C186 with 80287/80387/80C187 NPX)	$90
	Booter Toolkit (floppy disk bootstrap routines, DOS file system, light-weight multitasking, windows, fast memory management)	$85
Windows!	Cview Windows 3.0 Program Editor (multiple fonts, C formatting, SDK help, auto install)	$80
Windows!	WinMem (unlimited global memory handles for Windows 3.0, 4-byte overheadper block rather than 20, virtual memory for 286)	$80
	MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	$80
	COMM-DRV (robust serial communications driver; COM1-COM25; INT 14H & EBIOS; FOSSIL & multi-tasker compatible; inc. file transfer)	$75
NEW!	Software Tools for C (145 functions: TSR, EMS, interrupt handling, screen functions with screen generator, etc.)	$70
	XSPAWN (swap parent process to expanded memory or disk before running child)	$70
	Heap Expander (virtual memory manager using expanded memory, extended memory, and disk space)	$65
	Foundations-1 C++ Class Library (ASCII record I/O, bit arrays, exception handling, B-tree, persistent objects)	$60
	MultiStr (evaluates a string containing multiple ASCII arithmetic expressions with variables)	$60
	Coder's Prolog (Version 3.0; inference engine for use with C programs)	$60
	Async-Termio (Unix V compatible serial interface for MS-DOS; stty, ioctl, SIGINT, etc.)	$55
	Pascal P-Code Compiler & Interpreter or Pascal-to-C Translator (Level 0 ISO standard Pascal with some Level 1 features)	each $50
	CALC (ASCII algebraic expression evaluator, unlimited parenthesis nesting, symbols, 32 built-in functions, easily extended)	$50
	Backup & Restore Utility by Blake McBride (multiple volumes, file compression & encryption)	$50
	Floppy TAR (TAR backup and restore on MS-DOS devices; direct access to non-standard devices)	$50
	SuperGrep (exceptionally fast, revolutionary text searching algorithm; also searches sub-directories)	$50
	Window Boss & Data Clerk (windows, menus, data checking, on-line help, window drag, screen forms)	$50
	OBJASM (convert .obj files to .asm files; output is MASM compatible)	$50
NEW!	YATE (Yet Another Text Editor; written in C++, full-screen)	$50
	CLIPS (rule-based expert system generator, Version 4.3; advanced manuals available at additional cost)	$50
	NIH Class Library & Book (basic C++ classes *& Data Abstraction and Object-Oriented Programming in C++* in softback by Keith Gorlen)	$50
	Kier DateLib (all kinds of date manipulation; translation, validation, formatting, & arithmetic)	$45
	DES Encryption & Decryption (2500 bits/second on 4.77 MHz PC for on-the-fly encryption at 2400 baud)	$40
	FlexList (doubly-linked lists of arbitrary data with multiple access methods; specify C or C++)	each $40
	Virtual Memory Manager by Blake McBride (LRU pager, dynamic swap file, image save/restore)	$40
	Classy Forms (screen forms in both C and C++ versions)	$40
	RXC & EGREP (Regular Expression Compiler and Pattern Matching; RXC makes finite state machine from regular expression)	$35
	Bison & BYACC (YACC workalike parser generators; documentation; no restrictions on use of BYACC output)	$35
	PC-XINU (Comer's XINU operating system for PC)	$35
	Spell Pack (6 spelling programs, a hyphenator, and a 60K word list: Ispell, Microsp, Sp, Cspella, Spell, Hy, & the Dawg utilities)	$30
	REGX Plus (search and replace string manipulation routines based on regular expressions)	$30
	GNU Awk & Diff for PC (both programs in one package)	$30
Upgrade!	6-Pack of Editors (eight public domain editors for use, study & hacking; includes microEmacs 3.10, Stevie, and Elvis)	$30
Upgrade!	Crunch Pack (25 file compression & expansion programs)	$30
NEW!	Make Pack (eight versions of Make including GNU Make, Cake, and Gymake and a makefile maker)	$30
	CCALC (handy extended-precision calculator; real and complex models; many built-in functions)	$30
NEW!	String Pack (lots of string routines; C++ String class, BAWK, word wrap, fuzzy search, Boyer-Moore, Hypertext, etc.)	$25
	PC-MAIL (UUCP mailer by Wietse Z. Venema; send, receive, and manage UUCP mail)	$25
	PERL for MS-DOS (C, sed, awk, and shell all rolled into one language; includes hardcopy docs)	$25
	FLEX (fast lexical analyzer generator; new, improved LEX; BSD Version 2.3 with docs)	$25
	Using C++ Library (the code from the book by Bruce Eckel and then some; Zortech 2.0 compatible)	$25
	VAL Linker by David Troendle (a complete linker; excellent for learning about linking)	$20
NEW!	**Unix/386**	
	GNU Emacs Version 18.55 (includes complete source & executables)	$125
	Waffle BBS (interface to UUCP & B or C news; internal forums; files section; external processing; full USENET support)	$120
	GNU C Compiler (gcc) Version 1.37.1 (includes complete source & executables)	$100
	ViewComp Spreadsheet	$80
	GNU Debugger (gdb) Version 3.5 (includes complete source & executables)	$60
	GNU Smalltalk Version 1.1 (includes complete source & executables)	$50

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3464 USA

Voice: (512) 258-0785
FAX: (512) 258-1342
E-mail: info@acw.com

Free surface shipping for cash in advance **For delivery in Texas add 7%** **MasterCard/VISA**

Circle **22** on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Developing
C++
Applications?
Introducing...
Linnaeus
The C++ class browser and design tool

Graphical trees let you examine complex inheritance heirarchies, whether the classes are in one file or *hundreds* on multiple drives

Instant access to detailed information about every class on your machine - just click on its icon to view its data items, methods, comments, ancestors, descendents and filespec

Ad-hoc query lets you find *any* class or subset quickly

Interfaces with Brief

Full GUI environment - EGA/VGA required, mouse and EMS supported

$195

Also...
VMO Virtual Memory Objects

A dynamic memory structures class library. Objects can grow to four gigabytes using heap, EMM and disk. Transparently. Do things like:
MemCpy(obj2.obj1,10000000L);

Strings, arrays, lists, *m*-trees, b-trees, heaps, more. *Variable length* element sizes to *4gb*. Imagine a tree node that contains the text to *The Lord of the Rings*! Ideal for image manipulation and DBMS's

Included is our standard class library with extensive input, file, control, messaging, debug objects and simulated parameterized types

Over 70 classes, 1000 methods

$295

Zircel software
(303) 750-9543

library code can be shared by many different programs. It can also be updated without requiring every program that uses it to be recreated. A dynamic link library routine thus provides an easy upgrade path when newer versions are created; it simply replaces the library file in the system. This is certainly simpler than a complete relink of a program, which is required to upgrade static link library routines.

A second piece of code required to create a Windows program is the program start-up code. In particular, you will notice that Windows programs do not have a **main** function, but the **WinMain** function serves the same purpose. Before **WinMain** is called, several other functions are called. In particular, a routine to perform task start-up—**InitTask**—is called in the Windows KERNEL. A second routine, **InitApp,** is called in Windows' user-interface module (USER) to create a message queue and do other program start-up tasks.

Resources are read-only data objects that are created to help Windows run in a low-memory situation. In general, because most resources are read-only, they can be read into memory when needed and subsequently purged or "discarded" from memory when they are not needed.

If you take a close look at the different types of built-in resources, you'll see that many of them provide support for various user-interface objects. For example, resources are used to store menu and dialog box definitions as well as for various graphic objects such as bitmaps and fonts.

To bind a resource to a Windows executable file, you must use a tool called the resource compiler. Perhaps a better name for this utility would be *resource translator and linker*, which describes its actual functions. Whatever it is called, it is needed to create a Windows program. Of the compilers I reviewed, only one, the Borland C++ compiler, comes

Table 3: Required tools and their capabilities

Utility	SDK	SUPP
WINDOWS.H include file	✓	
Linker		✓
• Must be able to create Windows EXE		
• Must support module definition (DEF) file or its equivalent		
Static link libraries		
• Dynamic relocation records for Windows system calls (libw.lib)	✓	
• Windows application start-up code	✓	
Resource tools		
• Resource compiler	✓	
• Dialog box editor	✓	

Table 4: Standard prolog/epilog vs. the Windows prolog/epilog

Standard prolog/epilog	Windows prolog/epilog
	push ds ; place current DS
	pop ax ; in AX
	nop
	inc bp ; far-call flag
push bp	push bp
mov bp,sp	mov bp,sp
	push ds ; save caller's DS
	mov ds,ax ; install our DS
...Body of the function...	...Body of the function...
mov sp,bp	mov sp,bp
	pop ds ; restore caller's DS
pop bp	pop bp
	dec bp ;
ret [# bytes cleanup]	ret [# bytes cleanup]

Circle 23 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LISTING 1

```
/*   MIN.C   A Minimum Windows Program that displays a window.
 *           The window can be moved, sized, minimized and maximized.
 *           The WM_QUIT message is correctly handled so that the program
 *           terminates properly.
 */
#include <Windows.H>

/*   Function Prototypes       */
long FAR  PASCAL MinWndProc (HWND, WORD, WORD, LONG);

/*   Main Function:  WinMain   */
int PASCAL WinMain (HANDLE hInstance,   HANDLE hPrevInstance,
                    LPSTR  lpszCmdLine, int    cmdShow)
    {
    HWND     hwnd;
    MSG      msg;
    WNDCLASS wndclass;

    if (!hPrevInstance) {
        wndclass.lpszClassName = "MIN:MAIN";
        wndclass.hInstance     = hInstance;
        wndclass.lpfnWndProc   = MinWndProc;
        wndclass.hCursor       = LoadCursor (hInstance, "hand");
        wndclass.hIcon         = LoadIcon (hInstance,"snapshot");
        wndclass.lpszMenuName  = NULL;
        wndclass.hbrBackground = COLOR_WINDOW+1;
        wndclass.style         = NULL;
        wndclass.cbClsExtra    = 0;
        wndclass.cbWndExtra    = 0;

        RegisterClass( &wndclass);
        }
    hwnd = CreateWindow("MIN:MAIN",          /* Class name.   */
                        "Minimum",           /* Title.        */
                        WS_OVERLAPPEDWINDOW, /* Style bits.   */
                        CW_USEDEFAULT,       /* x - default.  */
                        0,                   /* y - default.  */
                        CW_USEDEFAULT,       /* cx - default. */
                        0,                   /* cy - default. */
                        NULL,                /* No parent.    */
                        NULL,                /* Class menu.   */
                        hInstance,           /* Creator.      */
                        NULL);               /* Params.       */

    ShowWindow (hwnd, cmdShow);

    while (GetMessage(&msg, 0, 0, 0)) {
        TranslateMessage(&msg);      /*  Keyboard input.      */
        DispatchMessage(&msg);
        }
    return 0;
    }

/*      Window Procedure:  MinWndProc  */
long FAR PASCAL MinWndProc (HWND hwnd,   WORD wMsg,
                            WORD wParam, LONG lParam)
    {
    switch (wMsg) {
        case WM_DESTROY:
            PostQuitMessage(0);
            break;

        default:
            return(DefWindowProc(hwnd,wMsg,wParam,lParam));
            break;
        }
    return 0L;
    }
```

LISTING 2

```
; MIN.DEF
NAME          MIN
EXETYPE       WINDOWS
DESCRIPTION 'Minimum Windows Program (c) 1990 By Paul Yao'

CODE          MOVEABLE DISCARDABLE
DATA          MOVEABLE MULTIPLE

HEAPSIZE      512
STACKSIZE     5000
EXPORTS
    MinWndProc
```

LISTING 3

```
# min.mak makefile for Microsoft C
min.res: min.rc min.cur min.ico
    rc -r min.rc
min.obj: min.c
    cl -AM -c -d -Gsw -Ow -W2 -Zpi min.c
min.exe: min.obj min.def
    link min,/align:16,/map, mlibcew libw/NOD/NOE/CO, min.def
    rc min.res
min.exe: min.res
    rc min.res
```

LISTING 4

```
; min.rc
snapshot icon min.ico

hand  cursor min.cur
```

LISTING 5

```
# makefile for MetaWare High C
min.res: min.rc min.cur min.ico
    rc -r min.rc
min.obj: min.c
    hc -c -HOn=Windows_exported -HOn=Windows_code -Mm min.c
min.exe: min.obj min.def
    link mini+metawin,/align:16,/map, libw mlibcew/NOD/NOE,
min.def
    rc min.res
min.exe: min.res
    rc min.res
```


WINDOWS APPLICATIONS IN 1/10th TIME!

* YOU CAN RAPIDLY PROTOTYPE FOR WINDOWS 3.0!
* Generates C Code for Complete SAA CUA User Interfaces.
* Does DATA VALIDATIONS and Spreadsheet-like TABLES.
* Ports In Minutes to DOS,Windows 3.0 and OS/2.
* Easy to Learn. Be MORE Productive In a few hours.
* FRONT ENDS to 3270/Other Hosts,LAN,SQL Server,Sybase, Informix,Oracle,BTrieve,dBVista,CIndex,Paradox,dBase....

Instant Prototyper. Both DOS & Windows,only $99.
PROTOTYPES:fancy,data oriented,50+Windows in one week.
ANIMATES:Windows/dialogs,Menus,Fields,Tables,Text File Editor,Help,Icons,radio/check/push buttons,pick/lists/combo boxes,Scroll bars,Bitmaps.VALIDATES:Custom/cross/lookup, display,required,picture(A9!LXB#Z$),range,date,time,money...

Instant Windows. Windows 3.0 $495.DOS $249.
Includes Prototyper,50+Function Library,350p Manual,C Code Generator (just add your DBMS/LAN/Internals) and examples.

Instant Windows Combo Pack. $995.
Includes Prototyper,Instant Windows for DOS & Windows,DLL, Linker,Dynamic Forms,250+Functions,1 Yr.Maintenance/Gold Level Tech.Support,4 hrs Premium Consulting,600p Manual.

CALL 415-324-9552 NOW! No Royalties.
No Runtimes.QUALITY Product,Tech.Support and Consulting.
WinSoft.1016 E.El Camino Real#216,Sunnyvale CA 94087.

Circle 24 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SLATE Universal Printer Driver

NEW! Text & Graphics!

By including SLATE, you can print both **Text and Graphics** on **over 400 printers. Immediately! Painlessly!**

You can use SLATE in your product with **no royalties.**

Make your product more functional and competitive by taking advantage of SLATE's **advanced text features:**

- Support multiple printers on the same system.
- Output to parallel printers, serial printers, DOS files, console, DOS print spooler, and Novell network printers.
- Support proportional fonts laser printer soft fonts.
- Set exact print positions.

You can add the **NEW** Graphics Subsystem for **advanced graphics features:**

- Print images from the screen, graphic files, or custom image systems.
- Scale and Rotate the printed image.
- Convert colors to patterns.
- Print grey scale (shaded) and color images.
- Intermix text and graphics.

What is SLATE?

SLATE is a set of C libraries with over 150 text printing functions, a Database of over 400 printers, and tools for end user configuration and testing.

SLATE with Graphics adds over 60 graphic printing functions.

Call now for more information. Order **SLATE for $299 or SLATE with Graphics for $448** with our risk free, 30 day return policy.

We accept Visa, Mastercard, COD's or PO's from qualified companies. Source code, maintenance, and site licenses are available. Also ask for information about our S_PRINT Text Formatting System for Software Developers.

800-346-3938

The Symmetry Group

PO Box 26195
Columbus, Oh 43226
614-431-2667
FAX 614-431-8734

Circle 25 on Reader Service Card

equipped with one. For the rest, you must purchase Microsoft's SDK.

TESTING FOR COMPATIBILITY

Testing a compiler for Windows compatibility involved checking whether I could take my minimum Windows program, MIN.C, and have a compiler get it into a form that was acceptable to Windows. To ensure that things were done correctly, I examined the code via various code disassembly utilities. I was pleased to find that several compiler vendors include just such a utility. JPI, for example, provides the TopSpeed DisAssembly utility (TSDA.EXE).

When I built MIN for each compiler, I ran the executable in all of Windows' operating modes (real, standard, and enhanced). After all, some of the low-level tricks are played for the benefit of real mode. There are also a few subtle differences between standard and enhanced mode, so this has become part of my standard test suites. The following pages contain vendor-specific notes that I put together during my tests of the Windows-compatible C compilers. The notes are organized by vendor, in alphabetical order.

BORLAND'S TURBO C++

Borland International got its start in the early 1980s by offering an inexpensive Pascal compiler through the mail. Phillipe Kahn, its founder, has built a credible software company through an assortment of acquisitions. Borland offers a number of language products, a spreadsheet, a database, and a popular DOS TSR program.

Of the compilers I tested, the one from Borland demonstrated the strongest commitment to supporting the development of Windows 3 applications. This is a paradox for the company that at one time planned to bypass Windows in favor of the OS/2 Presentation Manager (after all, Windows was just Presentation Manager's sidekick, wasn't it?). All puns aside, the folks at Borland seem to understand that Windows 3 is clearly different from earlier versions and that its current success is more than just a temporary fad.

The good news to programmers who have purchased version 2.0 of Borland's C++ compiler is that they don't have to purchase a Windows SDK from Microsoft. As the first participant in Microsoft's Open Tools Strategy, Borland has everything you need to develop Windows applications. This includes a Windows-compatible linker and debugger, all the needed library files, a customized version of the *WINDOWS.H* **include** file, and even a set of tools that replace Microsoft's for working with resources. Included in the resource tools is Microsoft's resource compiler (RC.EXE and RCPP.EXE) and a Resource Toolkit that proved to be the single disappointment in an otherwise great product.

This is one of two C++ compilers I tested (the other is the Zortech C++), but because my focus is on C code, I did not test this compiler with a C++ version of my test program. Instead, I used straight C.

The creation of MIN was simple: a single switch, **-W**, put in place the necessary prolog and epilog conventions. In fact, the compiler then called the linker, which was smart enough to read my program's module definition file. Here is the command line I used to build my program:

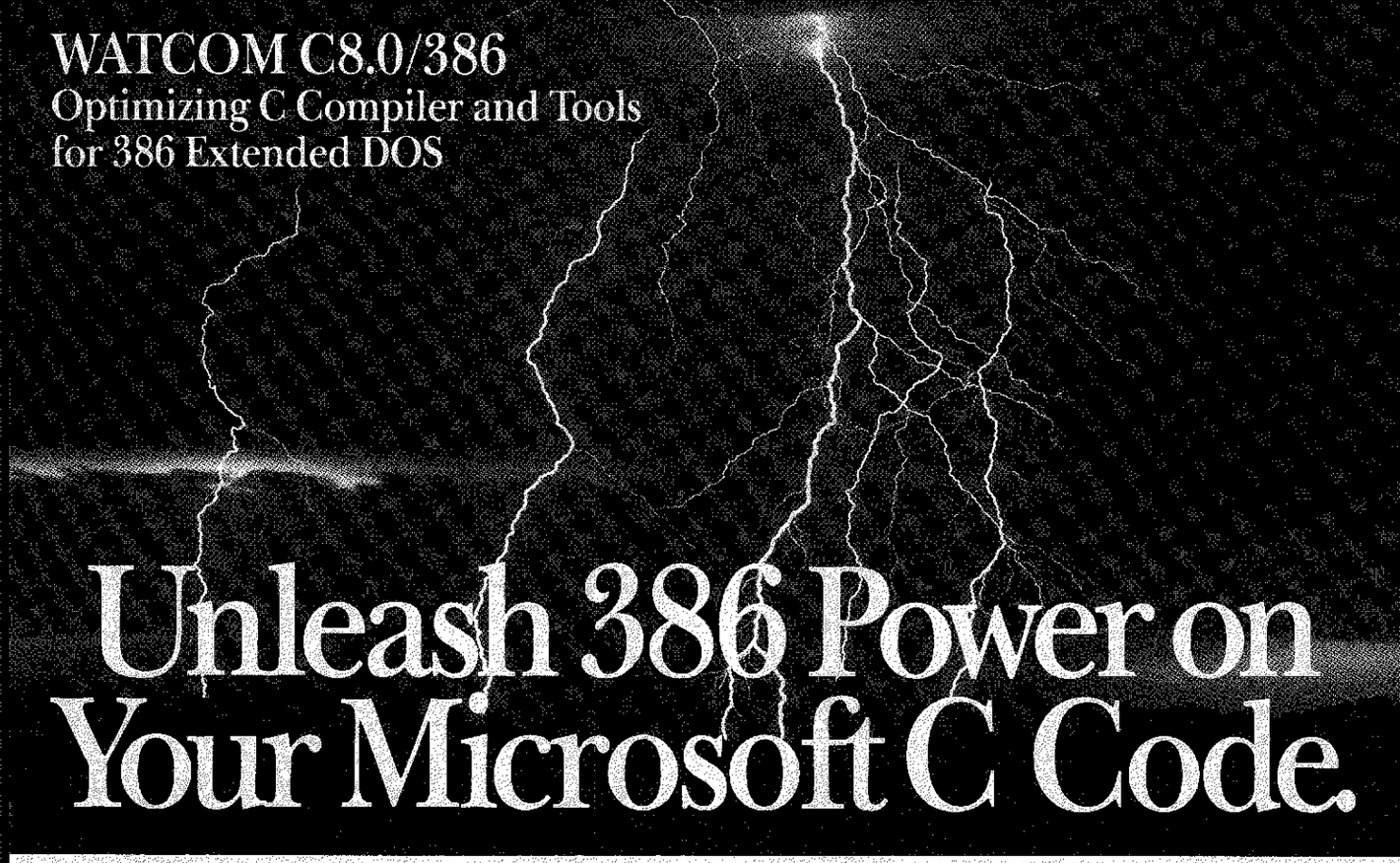
```
bcc -W min.c
```

That's it: no fuss, no muss. No changes were required to any of my program files. And no changes were required to *WINDOWS.H*, the Windows **include** file. The reason is simple: Borland includes its own version that has been fine-tuned for use with its compiler.

The Resource Toolkit program that Borland ships with this product was developed by the Whitewater Group, which created the Actor programming language. This program is functional, but I found it slow, quirky, and cumbersome. In particular, it doesn't provide what I would call an intuitive user interface. It took me quite a bit of effort to figure out how to use each of the individual tools in the resource toolkit. In all fairness, it is more useful than the tools provided in Microsoft's SDK if you're willing to take the time to learn to work around the quirkiness of this utility.

Aside from Microsoft (Windows SDK), Borland is the only vendor to provide a Windows-compatible debugger. Turbo Debugger for Windows (TDW) runs in Windows standard or 386-enhanced mode. It can support either

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



WATCOM C8.0/386
Optimizing C Compiler and Tools for 386 Extended DOS

Unleash 386 Power on Your Microsoft C Code.

- Interactive source-level debugger
- Generates high-performance code for 32-bit protected mode
- Microsoft source and library compatible
- Fast, tight code
- Profiler
- Protected-mode version of compiler
- Graphics library
- 100% ANSI C and SAA compatible
- Run-time compatible with WATCOM FORTRAN 77/386

Experts Agree on WATCOM C:

"When Novell went looking for a 32-bit compiler for use with the NetWare 386 developer's kit, the company selected WATCOM's...It's clear that Novell chose wisely; this product is a winner."
Fred Hommel, BYTE, December 1989

"WATCOM C/386 is a fantastic new ANSI C compatible compiler for 386-based PC's...If you have written your application in Microsoft C, you will love this compiler."
J. Richard Hines, Electronic Test, December 1989

"Microsoft library- and source- compatibility makes WATCOM C7.0/386 ideal for porting DOS applications to 32-bit native mode. This compiler enables full 386 performance without 640K limitations."
Richard M. Smith, President, Phar Lap Software, Inc.

"WATCOM is definitely the leader in object-level optimizations...For flat-out executable speed,...WATCOM C showed shining performance."
Computer Language, February 1989

WATCOM C8.0/386 Professional

• 100% ANSI C optimizing compiler • Protected-mode version of compiler • 386 run-time library object code • Windowed source level debugger • Profiler • Editor • 386 graphics library • MAKE • Linker • Object-code librarian • Object-code disassembler • Supports Phar Lap and ERGO DOS extenders

WATCOM F77/386

• 32-bit optimizing FORTRAN compiler based on WATCOM C technology • Full ANSI FORTRAN 77 plus extensions • Includes WATCOM C development tools: Windowed source-level debugger, Profiler, Editor, MAKE, Linker, Object-code librarian and Object-code disassembler • Protected-mode version of compiler • Run-time compatible with WATCOM C8.0/386 • Supports Phar Lap and ERGO DOS extenders

WATCOM **1-800-265-4555**

415 Phillip Street, Waterloo, Ontario, Canada N2L 3X2 Tel. (519) 886-3700 Fax (519) 747-4971

WATCOM C is a trademark of WATCOM Systems Inc. Trademarked names are the properties of their respective owners.
© Copyright 1990 WATCOM Products Inc.

Contact this advertiser directly.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

a single monitor using screen swapping, a second monitor, or remote debugging over a serial line. Features added in TDW include logging Windows messages, displaying memory (global and local heaps), and listing modules loaded.

JPI'S TOPSPEED C

Jensen & Partners International (JPI) develops and markets the TopSpeed C compiler. This four-year-old company is a Borland spin-off, and many of its programmers were involved with such successful Borland products as SideKick and Turbo Pascal.

I tested version 1.05b for Windows compatibility and was generally pleased with what I found. Like many other language compilers being sold today, the TopSpeed C compiler comes equipped with what JPI calls an *integrated development environment*. The TopSpeed Integrated Environment comes with a Wordstar-like editor, a project-organizing system, a built-in linker that can create DOS, Windows, or OS/2 programs, a *make* utility, and, of course, the compiler itself. The Visual Interactive Debugger (or VID) is also included with the compiler but unfortunately is not (yet?) Windows-compatible.

The version of the compiler that I tested came with a sample program, WDEMO. Compiling and testing this program convinced me that the compiler was indeed able to create a working Windows program. Creating my minimum Windows program, MIN, required a little more effort. The primary reason was the presence of one or two land mines that, once they're understood, are easily overcome.

One very interesting feature of this compiler is that it can create DLLs in DOS. Although you may be used to hearing about DLLs in Windows or in OS/2, this is the first product I have encountered that brings dynamic linking back to DOS. I have no doubt that the presence of this feature in the compiler helped JPI's developers make the move to Windows. Even though my tests don't include the creation of a Windows DLL, it's a safe bet that JPI's compiler will be able to create them.

One of the first things you must do when creating a program with the JPI compiler is to write a project file. This file is a list of switches that the compiler/linker uses to determine how to build an executable file. Here is the project file, MIN.PRJ, that I created:

```
make WinEXE
option +case +map +pack +check -vid +manual
option -codeview +windows
model Medium

export min
include nofloat
include initwin
import rs_wclib
import windows
run winpatch min.exe
run rc min.rc min.exe
```

A few of these lines are worth describing in more detail. The **include** statements describe the static link libraries that are to be made available to the executable file. Any code MIN needs from the **include**-type libraries will be copied completely into MIN.EXE.

The **export** statement caused me quite a bit of trouble, so I'll spend some time describing what it does, what went wrong, and how you can avoid spinning your wheels. In the world of dynamic linking, a module (an application or a DLL) can either import or export functions. When a program uses dynamically linked functions, it *imports* the required functions. To make a function available to other modules, a function is *exported*. Programs export call-back functions as part of the way they are hooked into Windows. Exported functions include window and dialog-box procedures.

JPI PROBLEM SOLVING

So far, so good. The problem occurs when you don't have a preexisting library to export. When this happens, the TopSpeed compiler tries to export the module you're trying to build. In my case, that meant MIN. While linking, I got several error messages for duplicate symbols. In effect, I was trying to recursively link—which is not something the linker was expecting!

Solving this problem is simple. You use the IMPLIB (*import library builder*) utility. As I mentioned at the start of this article, this utility is provided by

The Heap Expander™ v 2.05

Gain virtually unlimited heap space.

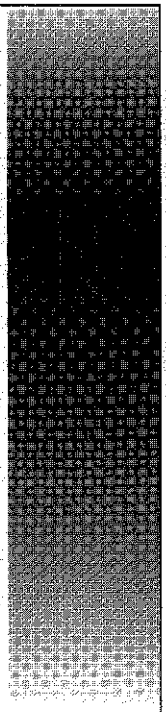
Function

Provides interface routines so that your programs use expanded memory, extended memory or conventional memory. As memory becomes full, The Heap Expander swaps to disk.

Utility

The Heap Expander comes with complete source code, libraries and documentation plus four utility programs that simplify development of expanded memory programs.

Includes libraries and source code for all current versions of Turbo C and Microsoft C.
System requirements: IBM PC, XT, AT or clone compatible with MS-DOS or PC-DOS v. 2.0 or above.



MC/VISA/COD
800-248-1045 ext 100 USA
800-952-5560 ext 100 Idaho

still just $79.95

THE TOOL MAKERS
P.O. Box 2151
Santa Cruz, CA 95063
408-458-0690

Circle 26 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

FREE ISSUE!

PJ readers are invited to receive a FREE sample issue of Midnight Engineering.

The Journal Of Personal Product Development provides unique mix of practical hardware and software engineering mixed with the business aspects of small scale product development. Your guide to developing and marketing your own products.

Midnight Engineering
Suite 7041
111 E. Drake Rd.
Ft. Collins, CO 80525
303-225-1410 (anytime)

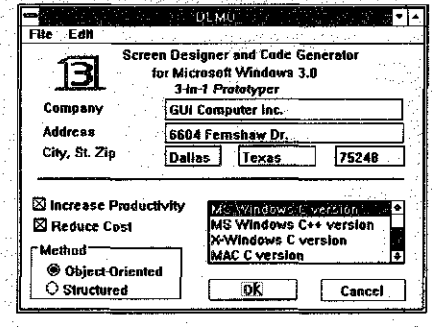
Circle 9 on Reader Service Card

C/C++ Windows 3.0

Screen Designer Code Generator

Only $99

The easiest way to learn and to master Microsoft Windows programming



With 3-in-1, developing this window screen requires only

20 Minutes !!

That includes screen layout, code generation, compiling and linking.

***GUI*'s 3-in-1** C $99 C++ $159 Both $199

30 Day Guarantee · No Source Code Royalties

Also many low cost Windows source code utilities.

1-800-800-9010 TEL 214-250-3472 Fax 214-250-1355

Contact this advertiser directly.

PC-lint presents C Bug # 571

```
#include <stdio.h>

main()
   {
   char ch = 0xFF;
   if( (unsigned) ch == 0xFF )
      printf( "equal\n" );
   }
```

What output does your C compiler produce and what warnings do you get? Is the output what you expected? If you can't find this bug, give us a call.

PC-lint will catch this and many other C bugs. Unlike your compiler, PC-lint **looks across all modules** of your application for bugs and inconsistencies.

"PC-lint, the one tool I believe *all* C programmers should have." -- *The C Gazette*

More than **270 error messages**. More than **90 options** for complete customization. Suppress error messages, locally or globally, by symbol name, by message number, by filename, etc. **Easy to Use** - If you know C you already know how to use PC-lint. Check for **portability** problems. Alter size of scalars. Adjust format of error messages. Automatically generate ANSI **prototypes** for your K&R functions.

New Features in Version 4.0

Lint Object Modules - incremental linting
Weak Definial Checking - macros, structs, typedef's, enums, and declarations not used globally and locally. Header files not used. Externals that can be made static. Declarations that can be moved from headers.

Mainframe and Mini Programmers

Get **FlexeLint** in shrouded source form for Unix, OS-9, VAX/VMS, QNX, IBM VM/MVS, etc. Requires only K&R C to compile but supports ANSI. Pricing starts at $798. Call for details.

Gimpel Software

3207 Hogarth Lane Collegeville PA 19426
(215) 584-4261 - CALL TODAY
(215) 584-4266 - FAX

PC-lint is only $139 - Specify MS-DOS or OS/2
PC-lint 386 is only $239 - requires MS-DOS on a 386
For power users with huge programs
30 Day Money-back Guarantee. PA add 6% sales tax.
PC-lint and FlexeLint are trademarks of Gimpel Software.

Contact this advertiser directly.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The habit of overlooking compiler errors means that more serious problems may be overlooked.

Microsoft with their C compiler and can also be obtained free of charge from Microsoft as part of the Windows SDK Supplemental Compiler Utilities Disk. IMPLIB reads a module definition (DEF) file and makes a copy of the functions listed in the EXPORTS section (see Listing 2 for MIN's DEF). Another problem worth mentioning is that the TopSpeed tools expect the exported names to appear in uppercase. It's a simple matter to change them so that, for example, the EXPORTS line in MIN.DEF reads as follows:

```
EXPORTS
    MINWNDPROC
```

The **import** statements provide the names of static relocation libraries. A static relocation library contains no code but instead lists the names of routines needed for a dynamic link. In this example, the *windows.lib* file is imported. This allows the linker to create the hooks needed to dynamically link three system DLLs to Windows: KERNEL.EXE, USER.EXE, and GDI.EXE.

The **run** statements list programs to be run as part of the program creation process. The *winpatch.exe* utility is provided by JPI to set some flags in the executable file's header to identify it as a Windows 3.0 program. Without this utility, Windows would refuse to load your program. The chief reason is that a single linker is used for DOS and OS/2. The flags in the EXE file header prevent Windows from running a program intended for another operating system.

The second **run** statement executes the resource compiler (RC.EXE), which converts the various user interface objects stored in MIN's resource file, MIN.RC. These are attached to the executable file, MIN.EXE, so that MIN's code, data, and user-interface data are all part of a single, monolithic file.

To incorporate the Windows prolog and epilog into a function, JPI relies on a set of pragmas. A pragma is a compiler directive; a *call* pragma sets up the prolog and epilog. Here's how to direct the JPI compiler to use Windows prolog and epilog for MIN's window procedure, **MinWndProc:**

```
#pragma save
#pragma call( windows = on )
long FAR PASCAL MinWndProc (HWND, WORD,  WORD,
                            LONG);
#pragma restore
```

The **#pragma save** and **#pragma restore** statements make sure the effect of the *call* pragma doesn't disturb other parts of our program.

The TopSpeed C compiler generated very small, fast code by making extensive use of the CPU registers. I found this out because the compiler comes with a very handy utility (TSDA.EXE) for disassembling the OBJ (and LIB) file(s). By using this tool, I could see an assembly-language version of any OBJ file—including those created by other compilers. This was useful for comparing the quality of the code generated by different vendors.

My only complaint with JPI's compiler is that quite a few warning messages are generated for near-to-far pointer conversion. This is described in one of the compiler's **read-me** files as being a matter of style, but I think more is involved. In general, warning messages should be reserved for situations in which programmer attention is required. Otherwise, if a programmer gets into the habit of overlooking compiler errors, there is a risk that a more serious problem may be overlooked.

In this particular case, the compiler has the function definitions (prototypes) from the Windows **include** file, *WINDOWS.H*. It also has the data structure definitions. Therefore, it should be able to convert near-to-far pointers without issuing a warning. As I see it, the primary problem is that a programmer has to create function casts to silence the bogus errors. This creates the additional risk that the cast will hide actual type-mismatch problems.

METAWARE'S HIGH C

Though it required the most effort of any of the tested compilers, the MetaWare compiler did indeed pass the test of being able to create Windows-compatible code. The effort that was required came as a surprise, in particular because MetaWare's compiler was the first non-Microsoft compiler to provide Windows compatibility.

As Mike Stebbins of MetaWare explained to me, MetaWare has focused its compiler development efforts on creating highly optimized code. MetaWare claims only "rudimentary support" for Windows. Even so, I was able to create a Windows 3.0 program with this compiler. Here are some of the things you'll have to do if you wish to use this compiler for Windows development.

The first problem I encountered was running out of symbol space. This happened because of the large size of the Windows **include** file, *WINDOWS.H*. When the compiler runs out of symbol space, it terminates with a "System Error" message.

Fortunately, MetaWare ships two compilers: a real-mode version and a 386 protected-mode version. Both produce 16-bit 8086 code, but the 386 protected-mode compiler uses the Phar Lap 386 VCPI DOS extender. Using this compiler, I had no problems with symbol space.

However, Phar Lap's current DOS extender is not compatible with the DPMI that Windows 3.0 uses. Therefore, compiling with the 386 protected-mode version of the MetaWare compiler required me to disable the HIMEM.SYS line in my CONFIG.SYS file. According to the folks at MetaWare, they will ship a new version of the compiler as soon as Phar Lap provides DPMI compatibility. Until then, HIMEM.SYS—and the Standard and Enhanced modes that it makes available—must be disabled for the 386 protected-mode compiler to operate.

The second item involved making changes to *WINDOWS.H* itself. Most of the compilers required one change or another. Here are four lines I added to *WINDOWS.H* to allow the MetaWare compiler to properly interpret the four keywords **near**, **far**, **pascal**, and **cdecl**:

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
#define cdecl
#define near _near
#define far  _far
#define PASCAL _CC(_CALLEE_POPS_STACK |
        _SAVE_REGS)
```

As described in MetaWare's documentation, certain changes are required to the definition of the FARPROC and NEARPROC typedef symbols. In addition, the definition of the WNDCLASS data structure must be changed. As you may know, this structure takes a pointer to a function. Because of the way the FAR and PASCAL keywords are defined,

```
LONG (FAR PASCAL *lpfnWndProc)();
```

has to be changed to:

```
LONG FAR PASCAL (*lpfnWndProc)();
```

The third set of things I had to do involved making changes to my source code itself. Like the JPI compiler, the MetaWare compiler will not automatically convert a **near** pointer to a **far** pointer when a function prototype or structure definition requires it. This requires a cast each time a pointer is passed as a parameter. In MIN.C, which is a relatively short program, 12 casts are required. Because explicit casts can hide type-mismatch problems, I am strongly opposed to them. On the other hand, leaving the cast out means having to wade through a large number of irrelevant warning messages.

CHANGES FOR METAWARE

Another set of changes that MIN.C required was the use of pragmas to establish the proper calling convention. This is discussed in MetaWare's documentation; here are the lines of code I included in MIN.C:

```
#pragma Calling_convention (_CALLEE_POPS_STACK |
                            _FAR_CALL |
                            _SAVE_REGS |
                            _WINDOWS);
far long MinWndProc (HWND, WORD, WORD, LONG);
#pragma Calling_convention ();

#pragma Calling_convention (_CALLEE_POPS_STACK |
                            _SAVE_REGS);
int WinMain (HANDLE, HANDLE, LPSTR, int);
#pragma Calling_convention ();
```

When I assigned **MinWndProc** (my window procedure) a pointer to the WNDCLASS element **lpfnWndProc**,

Program FASTER with ...

CC-RIDER

or the new

C++ RIDER

POP-UP SOURCE BROWSERS FOR ANY EDITOR!

CROSS-REF Editing!
Instantly pop-up your symbol definitions with a hypertext hot-key inside any editor! The Standard Edition has Source View, Edit and Paste commands. And the powerful *Professional Edition* even lets you walk through all uses of a symbol in your code as you edit!

Source Analysis!
Build a database of symbol information with our powerful ***source code analyzer*** which generates a wealth of useful output, like fully commented static and global function prototype #include files and a detailed log file showing WHERE and HOW all your symbols are used. And it uses expanded memory for ***lightning fast*** source code analysis!

QuickHelp, too!
CCSYM can also create a ***help database for your program*** which is compatible with Microsoft's help system, *QuickHelp*. All symbols in your application are accessible from QuickHelp's menus, with ***documentation extracted automatically from your original source code.***

Works with ANSI/Microsoft C, plus AT&T, Glockenspiel, Zortech, Borland C++. Supports Microsoft C v6 extensions.

FREE DEMO !

Standard	*Professional*
$89	***$279***
DOS or OS/2 C or C++	DOS & OS/2 C & C++

Western Wares
(303) 327-4898
Box C Norwood, CO 81423

Circle **27** on Reader Service Card

Turn your favorite C compiler into a powerful database manager with the

C/Database Toolchest

The **C/Database Toolchest™** adds sophisticated file management functions to your Power C™, Turbo C®, QuickC®, or Microsoft® C compiler. With the **C/Database Toolchest™**, your data requires much less disk space than with programs like dBASE®, and you can access your data much faster. Of course the full power of C provides you with an unlimited amount of programming flexibility.

The **C/Database Toolchest™** includes three major components:

1) An advanced B+tree library gives you instant access to your data.

2) A high-level ISAM library provides you with an easy-to-use C interface, and

3) A complete database manager (with C source code included) shows you how to create impressive applications.

You also receive a comprehensive 350 page manual and a utility for converting dBASE® files.

The **C/Database Toolchest™** supports features that you'd expect to find only in products costing ten times as much. Advanced features include variable length records, variable length keys, multiple keys per index, and multiple indexes stored in a single file. Your data files can contain an unlimited number of records, and each record can be as large as 32K bytes in length.

About the only thing that the **C/Database Toolchest™** doesn't do is cost you a lot of money. We've kept our price low so you can manage your budget as easily as your data.

Now Only $19.95!

Order now by calling our toll free number or mail the coupon to:

Mix Software
1132 Commerce Drive
Richardson, TX 75081

1-800-333-0330

60 Day Money Back Guarantee
Not Copy Protected ■ Royalty Free
For technical support, please call:
1-214-783-6001

Order Coupon

Name_____
Street_____
City_____
State_____ Zip_____
Telephone_____
Paying By____ Money Order____Check
____Visa____ MC ____ AX ____ Disc.
Card#_____
Exp. Date_____
Disk Size_____ 5¼_____ 3½"

Qty.	Product	Price	Subtotal
____	C/Database Toolchest	$19.95	_____
____	C/Database Library Source B+tree & ISAM library source code	$10.00	_____
	Add Shipping ($5 USA, $20 Foreign)		_____
	Texas Residents Add 8% Sales Tax		_____
	Total Amount of Your Order		_____

C/Database Toolchest and Power C are trademarks of Mix Software. QuickC and Microsoft C are registered trademarks of Microsoft. Turbo C is a registered trademark of Borland. dBASE is a registered trademark of Ashton-Tate.

Circle **28** on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

the MetaWare compiler complained. It wanted to enforce strict type checking and so declared this assignment to be illegal. To work around this, I created this typedef:

```
typedef long FAR PASCAL (*FARPROC2)();
```

and used it in the assignment of **lpfnWndProc**, as shown here:

```
wndclass.lpfnWndProc = (FARPROC2)MinWndProc;
```

The final changes to my source file involved creating the dummy symbol *__acrtused*:

```
extern int __acrtused;
```

This was necessary to get the compiler to link in the correct Windows start-up code. The MetaWare optimizer noticed I hadn't used this variable and wanted to omit any reference to it in the OBJ file. I had to trick it with the following in my **WinMain** function:

```
if (__acrtused);
```

When the MetaWare compiler creates its object files, it inserts a symbol for the memory model. Because I always use the medium memory model, it inserted the MEDIUM? symbol. This helps to prevent object modules created with different memory models from being (incorrectly) put together. Unfortunately, because I was using the Microsoft and not the MetaWare libraries (a practice that MetaWare explicitly recommends), I received an unresolved symbol message for MEDIUM?. Resolving this required me to write the following small assembly language program. Once assembled and linked with the output of the MetaWare compiler, MIN.EXE was ready to run:

```
_MWLITERALS segment word common 'data'
public MEDIUM?
MEDIUM?  db 0
_MWLITERALS ends
```

The only other thing you'll need to have with the MetaWare compiler is a copy of the **make** file I used (as shown in Listing 2). As with the other compilers, I used Microsoft's MAKE program because it could recognize the MIN.MAK I was using. If you are currently a user of MetaWare's High-C compiler, you'll be happy to know that it can be used to create Windows programs. Hopefully, my list of instructions detailing what you need to do will be complete enough to get you started on your way.

WATCOM C 8.0

Watcom Systems, Inc. is an 18-year-old Canadian company that has developed language compilers and interpreters for the APL, BASIC, C, COBOL, Fortran, and Pascal languages. It has created these language products on a number of hardware platforms, including the IBM 370, the DEC VAX, and IBM-PCs and their compatibles.

I tested version 8.0 of the Watcom C compiler, using the "Professional Edition." Out of all the compilers I tested for this article, Watcom's proved to be more compatible with Microsoft's C compiler than most. This primarily means that building MIN was easier. No changes were required to the source code in MIN.C.

Watcom provides a choice of two compiler programs: WCL.EXE, which compiles and links, and WCC.EXE, which only compiles. I chose to use WCL.EXE, because it allowed me to choose a set of compiler switches that were exactly identical to Microsoft's. I used this command line to create MIN:

```
wcl -mm -c -s -zw -oas -w2 -zpi min.c
```

The role of each switch is described in Table 5.

One difference between the Watcom and Microsoft development tools is that Watcom's linker, WLINK.EXE, needs a special command file. Watcom provides a utility, MS2WLINK.EXE, to create this file from a Microsoft linker command line and a Microsoft module definition (MIN.DEF) file. Here is the Watcom linker command file I used:

```
#
#  Directive file for WATCOM linker, wlink.
#

file min.obj
library windows.lib
library clibm.lib

format windows

segment class 'code' moveable discardable

option map
option nodefaultlibs
option alignment=16
option modname=MIN
option manyautodata
option heapsize=512
option stack=5000
export MinWndProc
```

The *windows.lib* library referenced in this command file allows the linker to create dynamic-link relocation records for the Windows 3.0 system calls. It serves the same function as the *libw.lib* library from which it is created. Watcom provides a batch file, *windows.bat*, that converts *libw.lib* into *windows.lib* while modifying its floating-point support for better Watcom compatibility. At the same time, the Windows SDK floating-point library, *win87em.lib*, is also updated.

The *clibm.lib* file contains a special Windows version of Watcom's C runtime library. My only complaint with Watcom's product is the lack of a sample Windows program. In many respects this is a minor point, because even Microsoft does not provide a Windows sample with its compiler. However, given that a different linker input file is required, a sample Windows program would have been very helpful in getting MIN up and running. Hopefully, you can learn from my experience if you're a Watcom compiler user.

ZORTECH C++

I tested the Zortech C++ compiler for its ability to create Windows 3.0 programs. As with Borland's C++ compiler, I didn't write any C++ code but rather used the C-compiling capabilities of this compiler. It performed well and didn't require any changes to either my source code or to the Microsoft **include** file, *WINDOWS.H*.

The Zortech compiler was also able to create code that could be used with Microsoft's Codeview debugger. The primary reason is that Zortech's linker, BLINK, does not create Windows-compatible code. Instead, it relies on Microsoft's linker to do so. In this, though, it demonstrated an ability that

Table 5: Watcom switches

mm	Medium model
c	Compile only; do not link
s	Inhibit stack checking
zw	Enable Windows prolog and epilog
oas	Relax aliasing and optimize for size
w2	Warning level 2
zp1	Pack data structures to a byte alignment

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

was lacking in the other compilers: it could create OBJ files that were recognizable by Microsoft's linker.

Users of the Zortech C compiler must also rely on the static link libraries provided in Microsoft's SDK. Although Zortech provides their own *make* utility, I used Microsoft's to control the program-building process. In short, the only tool I used from Zortech was its compiler, ZTC.EXE. The command-line options I used are shown in this **make** file:

```
min.res: min.rc min.cur min.ico
    rc -r min.rc

min.obj: min.c
    ztc -mm -c -W -g -o+space -a -br min.c

min.exe: min.obj min.def
    link min,/align:16,/map, mlibcew
    libw/NOD/NOE/CO, min.def
    rc min.res

min.exe: min.res
     -rc min.res
```

Here the **-mm** switch requests the creation of a medium-model program. The **-c** switch specifies compile only, no link; **-W** requests the Windows prolog and epilog conventions. Symbol information for use with a debugger is generated by **-g**. Optimization for space is enabled by **-o+space**, and the **-a** switch specifies that data structures should be aligned on a byte boundary.

The **-br** switch calls for the protected-mode version of the compiler to run. This is necessary to avoid the problems created by the very large **include** file, *WINDOWS.H*. Without this option, the real-mode version of the compiler will run out of heap space.

Even with this option, I was unable to run the compiler under Standard Mode Windows—my only option was to shut down Windows completely and run the compiler by itself. Because it complies with the DPMI standard that Enhanced Mode Windows makes available, this option can be run under that mode instead.

If you're tooling up to create Windows programs, you'll be pleased to know that the majority of vendors have already adapted their compilers to generate Windows-compatible code. This may require that you upgrade your compiler, but if you've been thinking about upgrading this may be just the reason you need.

VENDOR INFORMATION

Turbo C++ 2.0
Borland International, Inc.
P.O. Box 660001
Scotts Valley, CA 95066-0001
(800) 331-0877
Circle 211 on Reader Service Card

TopSpeed C
Jensen & Partners Intl. (JPI)
1101 San Antonio Road, Suite 301
Mountain View, CA 94043
(415) 967-3200
Circle 212 on Reader Service Card

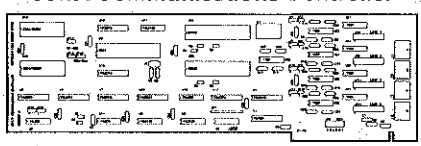
High C
MetaWare, Inc.
2161 Delaware Avenue
Santa Cruz, CA 95060-5706
(408) 429-6382
Circle 213 on Reader Service Card

C 6.0 and Windows SDK
Microsoft, Inc.
One Microsoft Way
Redmond, WA 98052
(800) 426-9400
Circle 214 on Reader Service Card

Watcom C 8.0
WATCOM Systems, Inc.
415 Phillip Street
Waterloo, ON, Canada N2L 3X2
(800) 265-4555
Circle 215 on Reader Service Card

Zortech C++ 2.1
Zortech, Inc.
4-C Gill Street
Woburn, MA 01801
(800) 848-8408
Circle 216 on Reader Service Card

Intelligent Four Channel
Serial Communications Controller



- 8031 microcontroller
- 82530 SCCs
- 32K memory
- RS-232
- RJ-45 connectors
- DOS device driver
- FCC Class A approved
- Excellent documentation
- BASIC, C, FORTRAN, and Pascal software source examples
- SCO Xenix drivers available
- **$495 complete** with all cables, software, & extensive documentation

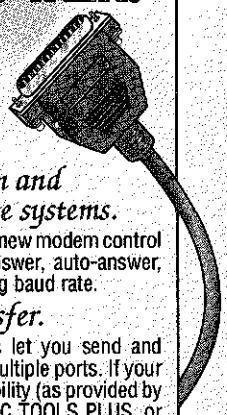
Designed, Manufactured, Sold and Serviced by:

RTech Inc.
Voice . . . (612) 345-4555
FAX . . . . (612) 345-5514
Modem . . (612) 345-4656
907 N. 6th St., Lake City, MN 55041

Circle **52** on Reader Service Card

WE HAVE WAYS TO MAKE YOU TALK.

(And let you listen in, too)



with tools from Blaise Computing.

Talk to your modem and connect with remote systems.

Go online with ease with our new modem control routines. Initialize, dial, answer, auto-answer, and auto-adjust for incoming baud rate.

Let's talk file transfer.

XModem/YModem routines let you send and receive multiple files over multiple ports. If your program has ISR/TSR capability (as provided by our POWER TOOLS PLUS, C TOOLS PLUS, or Turbo C TOOLS), you can talk file transfer in the background while your foreground process talks data input, or database query, or whatever.

The file transfer capabilities include 1K packets, CRC error checking, YModem (multi-file transfers with file name and size preserved), automatic adaptation to incoming packet size and error detection method. And we're talking fast — all file transfer routines have been optimized for maximum throughput.

Let's talk basics.

C ASYNCH MANAGER™ and **ASYNCH PLUS™** retain the features which have made them the libraries of choice for asynch projects — buffered interrupt-driven input and output to multiple COM ports with speeds up to 19200 Baud, XON/XOFF protocol, hardware handshaking and much more.

C ASYNCH MANAGER supports Microsoft C, QuickC and Turbo C. ASYNCH PLUS supports Turbo Pascal and QuickPascal. Both products require DOS 2.00 or later and an IBM PC, XT, AT or PS/2 machine or true compatible.

Let's talk price.

Why pay more? Our ASYNCH packages are just $189, including complete source code, a comprehensive reference manual with extensive examples, sample programs and online help.

Turn that serial cable into a party line with View232!

Debugging serial communications doesn't have to leave you talking to yourself. **View232™** turns your computer into a serial data analyzer that lets you listen in on the conversation between any two serial devices — and this party line provides a transcript of what was said!

View232 displays the data as it flows in both directions. Save a whole transmission in a buffer, then browse through or search the buffer for a pattern or specific character. And you can save the data to disk or print it for later study. VIEW232 is easy to use. And we supply the cable, all for just $189!

Let's talk track record.

Blaise Computing has produced a collection of tools that are unsurpassed for reliability, flexibility, and ease of use.

And, we're talking guaranteed.

If during the first 30 days you're not completely satisfied, we'll refund your money.

Call (800) 333-3087 today!

BLAISE COMPUTING INC.
2560 Ninth Street, Suite 316
Berkeley, CA 94710
(415) 540-5441
FAX (415) 540-1938

Trademarks are property of their respective holders

Circle **29** on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Fortran 90 is coming!

New features to look for in the upcoming Fortran 90 Standard

By Thomas M. Lahey

As FORTRAN 77 was receiving its final touches in 1978, the members of X3J3 (the ANSI committee responsible for Fortran standardization) were already organizing for the next effort. These Fortran devotees recognized that FORTRAN 77 was missing some important features. Among these were: some global declarations, such as INCLUDE and INTERFACE BLOCK; additional control structures; syntax and semantics for array operations; and improved numerical computation facilities, such as typing an object by expressing its range and/or its precision.

After completing FORTRAN 77, the group planned to standardize the above items, hoping to finish around 1983. The work of that pioneering group, their successors, and additional members who have joined over the years is finally coming to a close. Fortran 90 has now completed the third (and probably the last) U.S. public review.

WG5, the world group responsible for Fortran at the international level, is meeting in London in March to finalize the draft they will recommend for adoption. Every effort is being made to ensure that the U.S. Standard and the International Standard will be the same.

This article provides an overview of the changes in Fortran 90. I will discuss some of the major features that Fortran 90 standardizes, with an explanation of their use. I have carefully tried to avoid errors but lack a Fortran 90 language system to verify the presented material.

There are four known areas of potential conflict between FORTRAN 77 and Fortran 90: intrinsic naming conflicts; formatted input lists; DATA statement constant conversion; and SAVE attributes. The intrinsic subprogram names that Fortran 90 introduces may conflict with user-defined names in an existing program. To avoid this problem, use the EXTERNAL statement to list the subprograms required by a program unit.

In FORTRAN 77, the formatted input list requires either the same or fewer values as those of the record. Fortran 90 specifies that a record be padded with blanks unless PAD= 'NO' is specified in the OPEN statement for the unit from which the record is being read.

In the following FORTRAN 77 example, the 1.37 may be converted as if it were a DOUBLE PRECISION value (this conversion is compiler-dependent), whereas Fortran 90 processes the value as single precision. Note that both standards state that the type of a constant is declared by its syntax.

```
DOUBLE PRECISION D
DATA D/1.37/
```

In Fortran 90, local variables are scalars and arrays that are known only to a program unit. They may be passed to invoked subprograms from within that program unit. In a DATA statement, local variables have the SAVE attribute. This is compiler-dependent in Fortran 77.

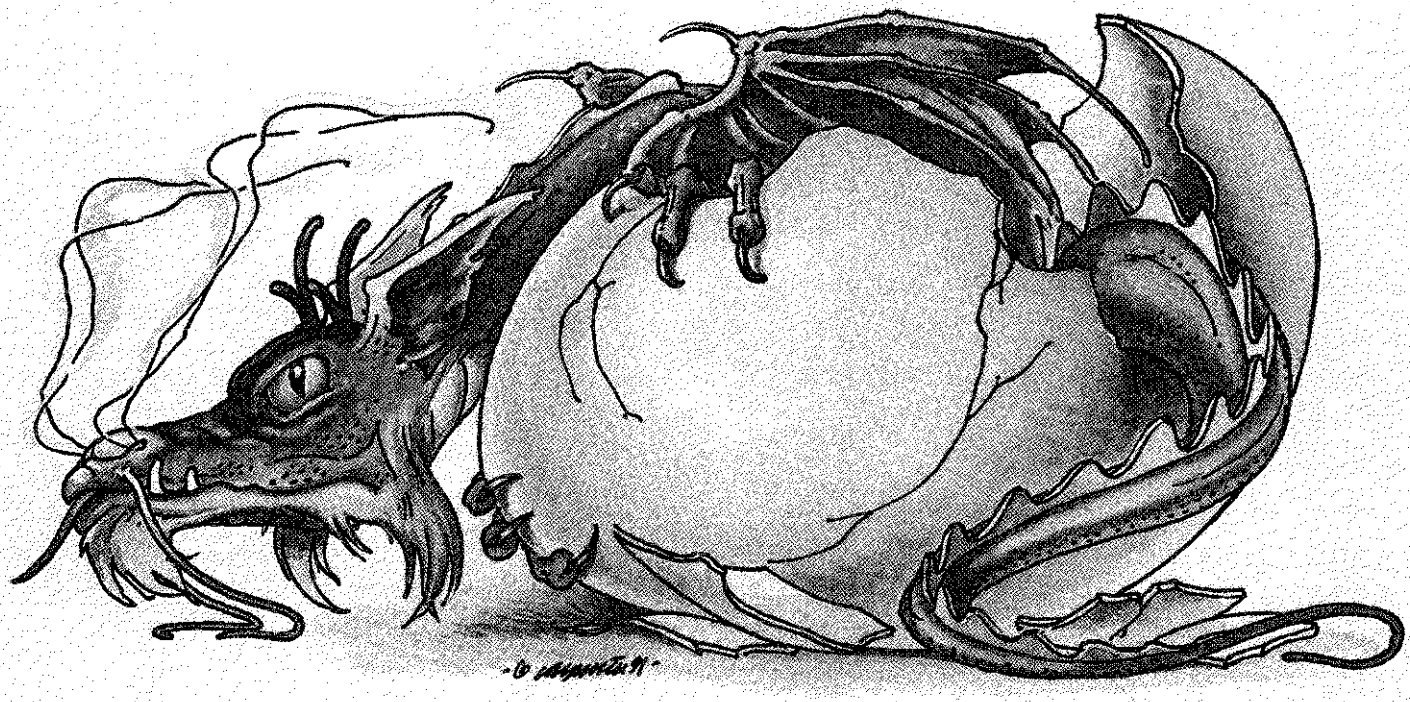
SOURCE FORM OPTIONS

The extended character set includes lowercase letters and the characters <, >, ?, ;, %, !, &, and ". The semicolon can separate multiple statements per line and the quote is a synonym for the apostrophe. The empty string ('' or "") denotes a string of blanks of the length required to satisfy its context.

Identifiers are no longer limited to 6 characters and can now be up to 31 characters in length with embedded underscore characters. Mixing upper and lower case is permitted in a name, but names are not case-sensitive. This means that compilers and other tools will not preserve the case of program names (they are mapped to the same literal), possibly resulting in some loss of programmer documentation.

Thomas Lahey is president of Lahey Computer Systems, developer of Fortran language systems for the PC and other environments. He has been involved in developing the Fortran 90 Standard. He may be reached at (702) 831-2500 or Lahey Computer Systems, P.O. Box 6091, Incline Village, NV 89450.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Two source forms are allowed. The *fixed source* form (columns) continues to be supported; blanks are not significant. The new standard introduces a *free source* form for expressing a Fortran program. A source line may contain from zero to 132 characters, and blanks are significant. Statement continuation is denoted by "&" as the last character on a line (the "&" cannot terminate a trailing comment, and continuing a broken token is more complex). Blanks are significant in *free source* form, so the following is not allowed:

```
do10i=1,3
```

Instead, the code must be

```
do 10 i=1,3    ! a trailing comment
```

Blanks are optional within a list of 18 keywords, including GOTO, ENDIF, and ELSEIF. They are required after a list of eight keywords, among them IMPLICIT, CASE DEFAULT, and *type-spec* FUNCTION. Synonyms for relational operators have been added (see Table 1).

TABLE 1: Relational operators

FORTRAN 77	Synonyms	Meaning
.LT.	<	less than
.GT.	>	greater than
.NE.	/=	not equal to
.EQ.	==	equal to
.LE.	<=	less than or equal to
.GE.	>=	greater than or equal to

Either form supports inline commands. When the exclamation point is not in character-constant context, it denotes that the remaining portion of the line is a comment. Watch out, though—an exclamation point in any column but six in fixed-source form denotes a comment for the remaining portion of the line, but in column six it denotes continuation.

Finally, in addition to source and comment lines, Fortran 90 has an INCLUDE line of the form INCLUDE *char-literal-constant*. This is the INCLUDE statement currently supported by most compilers. Note that the processor must support nesting of INCLUDE files.

ATTRIBUTES

The concept of *attribute* has been introduced in Fortran 90. One or more attributes may be declared in a type or derived type statement:

```
REAL, DIMENSION (366,24), SAVE :: temps, pressures
```

The double colon separates the attribute and variable lists and is required if more than one attribute or initialization is specified. Initialization uses this new declaration syntax:

```
REAL, DIMENSION (5, 3) :: a = 0.0
```

Note that this syntax is consistent with array assignment, as discussed below.

All of the attributes in the attribute list are applied to all of the names in the statement. There is a statement for every attribute. Table 2 lists the attributes and their meanings.

X3J3 examined questions such as how to allow for Kanji characters, how to port code with 64-bit REALs to a PC, and how to standardize COMPLEX type with greater precision than REAL type.

The members of the group solved these problems (and more like them) by parameterizing type declarations with KIND= and then adding a set of **intrinsic functions** to permit KIND determination and manipulation at compile time. Note in the following examples that multiple statements per line are used:

```
COMPLEX ( KIND=KIND(0D0) ) dc;  REAL ( KIND=KIND(0E0)  ) r
PARAMETER (kanji = 2);  CHARACTER (LEN=10, KIND = kanji)  msg
```

In these examples, *dc* is a DOUBLE PRECISION COMPLEX value, *r* is a REAL variable, and *msg* is a CHARACTER

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

variable of the second kind. The compile-time **intrinsics** associated with KIND are:

- SELECTED_REAL_KIND(p, r): This will return the KIND number that satisfies precision, *p*, and range, *r*; *p* and *r* are INTEGERs.
- SELECTED_INT_KIND(r): This returns the KIND number that satisfies range, *r*; *r* is an INTEGER.
- LOGICAL(bool, KIND): Here, LOGICAL variable *bool* is converted to the LOGICAL KIND and returned.

The syntax to include kind in numeric and logical constants is *constant*_**kind** (substitute the data type of *constant*). For examples:

```
1.0_SELECTED_REAL_KIND(18,28)
.TRUE._BIT
```

TABLE 2: Attributes and their meanings

PARAMETER	Define and type PARAMETERs of a type in a statement.
access-spec	Either PUBLIC or PRIVATE, appearance limited to MODULEs. This marks whether or not names are exported.
ALLOCATABLE	Array size and allocation occur at execution time. The number of dimensions (rank) must be declared.
DIMENSION	The parenthetical dimension declaration follows this attribute.
EXTERNAL	Due to the type attribute, list names are functions.
INTENT	Followed by a parenthetical declaration of IN, OUT, or INOUT; the variable list is limited to dummy arguments. This improves diagnostics and optimizations.
INTRINSIC	As in FORTRAN 77.
OPTIONAL	List variables are arguments that might not appear in the invoking program unit argument list. A LOGICAL INTRINSIC, PRESENT(arg), is available to test if an invoking subprogram supplied the arg.
POINTER	List variables must be associated by executing either a POINTER assignment or an ALLOCATE statement before the variable can be defined or referenced.
SAVE	As in FORTRAN 77.
TARGET	List variables may have a POINTER associated with them. This attribute permits optimizations.

TABLE 3: Intrinsics for KIND determination

DIGITS(x)	Number of significant digits
EPSILON(x)	Smallest REAL that EPSILON(x) +1 is not 1
HUGE(x)	Largest REAL
MAXEXPONENT(x)	Maximum exponent
MINEXPONENT(x)	Minimum exponent
PRECISION(x)	Decimal precision
RADIX(x)	Base
RANGE(x)	Decimal exponent range
TINY(x)	Smallest REAL

Here, BIT is a PARAMETER. For CHARACTER constants the syntax is **kind_'constant'** or **kind_"constant."**

Binary (B), octal (O), and hexadecimal (Z) constants are available, in the form:

```
B'...',  O'...',  Z'...'
```

where ... is a sequence of one or more of 0 or 1 for B; 0–7 for O; and 0–9, A–F for Z.

The compile-time **intrinsics** in Table 3 facilitate parameterizing a program from a numerical-analysis point of view. Note that these function results are based on the KIND of the argument *x*.

DERIVED TYPES

Fortran 90 has structures! The following example shows nested-derived types (*name* in **person**); a structure constructor with **person('John', ...)**; and a component name—**employees(1)%who%mid_initl**. Note that the % delimits structured components.

```
TYPE name
     CHARACTER*16 first
     CHARACTER*1  mid_initl
     CHARACTER*32 last
END TYPE name

TYPE person
     TYPE (name)  who
     CHARACTER*10 tele
END TYPE person

TYPE (person), DIMENSION (500), SAVE :: employees
employees(1) = person('John', 'Q', 'PUBLIC', '8005551212')
WRITE(*,*) employees(1)%who%mid_initl      !output is Q
...
```

DYNAMIC MEMORY ALLOCATION AND POINTERS

The ALLOCATE statement dynamically allocates space to variables or arrays. An example is shown in Listing 1. Note that the example code results in the largest application within a power of two, but not the largest in absolute terms.

Fortran 90 has POINTERs that behave like scalar and array dummy arguments: POINTERs can be typed and dimensioned but not invoked. This new data type eases creating linked lists, trees, and other powerful data structures. A POINTER is in one of three states, as shown in Table 4.

A LOGICAL **intrinsic function,** ASSOCIATED (*pointer*, *target*), where *target* is an OPTIONAL **arg,** returns .TRUE. in three situations: when *target* is missing and *pointer* is associated with a TARGET; when *target* is present, is a declared TARGET, and *pointer* is associated with it; and when *target* is present, is a POINTER, and *pointer* and *target* are associated with the same declared TARGET.

An example is given in Listing 2. Alternatively, the programmer could avoid the declaration of *array1* and *array2* by using *pntr1* and *pntr2*:

```
ALLOCATE ( pntr1(50,50), pntr2(70,90) )
```

Note that POINTERs may also appear in a DEALLOCATE statement.

ARRAY OPERATIONS

A small but growing amount of scientific computation is accomplished on vector and array processors. Fortran 90 meets

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LISTING 1

```
      REAL, DIMENSION(:,:), ALLOCATABLE :: temps, pressures
      ...
      n = 16384              ! Try for arrays of 24*8192 elements
10    n = n/2                ! n too big, halve it
      ALLOCATE ( temps(24, n), STAT=notice )
      IF ( notice .NE. 0 ) GO TO 10    ! temps not allocated
      ALLOCATE ( pressures(24, n), STAT=notice )
      IF ( notice .NE. 0 ) THEN
         DEALLOCATE ( temps );  GO TO 10
      ENDIF

!  pressures and temps have been allocated 24 by n elements.
      IF ( n < 1024 ) THEN
         PRINT '(" Only able to allocate"I4," elements")', n
         STOP "Quitting"
      ENDIF
      ...
```

LISTING 2

```
!  pntr1 & pntr2 associate only with two-dimensional REAL arrays
      COMMON /pointers/ pntr1, pntr2
      REAL, POINTER, DIMENSION(:,:) ::  pntr1,  pntr2

!  array1 & array2 are descriptors that "know" they are unallocated
!  TARGET is required since they will be associated with a pointer
      REAL, TARGET, DIMENSION(:,:)  ::  array1, array2
      ...
      ALLOCATE ( array1(50,50), array2(70,90) )
      pntr1 => array1;  pntr2 => array2  !POINTER ASSIGNMENTs
      CALL s          ! if s declares COMMON /pointers/, then it can
                      ! access array1 and array2
      ...
```

LISTING 3

```
      FUNCTION elements(string)            ! Count words
      IMPLICIT NONE;          INTEGER i
      CHARACTER*(*) string;  LOGICAL separator
      TYPE inventory
         INTEGER nwords, nletters, npunct, nblanks, nelse
      END TYPE inventory
      TYPE (inventory) elements

!  Initialize structure, INTRINSIC TRIM removes trailing blanks
      elements%nwords = 0;  elements%nletters = 0
      elements%npunct = 0;  elements%nelse = 0
      elements%nblanks = LEN(string) - LEN( TRIM(string) )

      IF ( string == '' ) RETURN                ! All blanks
      separator = .TRUE.                        ! To count words

  block1:  DO  i = 1, LEN( TRIM(string) )  ! No trailing ' '
               SELECT CASE ( string(i:i) )
                  CASE ( ' ' )                  ! Blank
                     elements%nblanks = elements%nblanks +1
                     separator = .TRUE.
                  CASE (a:z, A:Z)               ! Letters
                     elements%nletters = elements%nletters +1
                     IF ( separator ) THEN    ! New word?
                        nwords = nwords +1  ! Yes
                        separator = .FALSE.
                     ENDIF
                  CASE ( '.', ',', ';' )        ! Punctuation
                     separator = .TRUE.
                     elements%npunct = elements%npunct +1
                  CASE DEFAULT                  ! All others
                     elements%nelse = elements%nelse +1
               END SELECT
         END DO  block1
      END
```

LISTING 4

```
      MODULE ISO_string

!  Derived-type dynamic-length CHAR item: POINTER to rank-one array
      TYPE string          ! User defines DERIVED-TYPE STRINGs
         PRIVATE           ! Component "chars" unavailable to user
```

continued

MORE 34010 Video Co-Processors from PC Tech

- Mono 736 x 1024 - 2048 x 1536
- Color 1024 x 768 (1280 x 1024 soon!)
- 512K - 4M DRAM
- Optional CCITT & DGIS support
- **AGE XoftWare X Window Server certified board**
- C Compiler, Assembler, Linker, Debugger, DGIS Development Kit & TIGA Toolkit available
- **TIGA** included with all boards
- Custom designs available
- **PC Tech understands the 34010**

For further information, please contact:

PC Tech
907 N. 6th St.
Lake City, MN 55041
(612) 345-4555 (Voice)
(612) 345-5514 (FAX)

PC Tech is a trademark of PC Tech, Inc.

PC, XT, AT, DGIS, XoftWare, and DGIS are trademarks or registered trademarks of their respective companies.

Circle 30 on Reader Service Card

UNIX VERSION NOW SHIPPING

Experts Always Agree on the Solution!

However, they rarely agree on the features that make Salford's FTN77/386 superior!

"The defaults for the compiler are easily configured, and the compiler supports a load and go option for quick testing".
. . . Steve Baker - "Programmer's Journal ", July 1990

"The debugger was very easy to use. There is an on-line help facility which made reference to the manual unnecessary".
. . . Mike Gunn and Arul Britto, ".EXE", May 1989

"FTN77/386 provides more intrinsic functions than any other: functions to manipulate string, handle screen and keyboard I/O, read the date and time, parse the command line . . . Advanced file manipulation functions allows an FTN77/386 program to work with files in a C-like manner".
. . . Scott Ladd, "Computer Language", Nov. 1988

"For sheer speed and sleekness, the OTG compiler stands out like a Ferrari among Yugos".
. . . Jack W. Crenshaw, "Computer Language", May 1990

•Important Benchmark Results•

73% - of our new business in 1989 came through existing customer references
99.6% - satisfaction rating from our existing accounts
100% - guaranteed satisfaction

OTG SYSTEMS INC

Clifford, PA • (717) 222-9100 • Fax: (717) 222-9103

SERVICE, QUALITY AND INTEGRITY

FTN77/386 Developed by Salford Software Marketing

Circle 31 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Array operations in Fortran 90 are element-by-element, not algebraic.

this increasing demand by specifying array expressions and a supporting set of **intrinsic functions**. The definitions are:

- **Rank**: The number of array dimensions. A scalar has zero.
- **Extent**: The size of one dimension of an array (for example, 0:3 is 4).
- **Shape**: The rank and extents. This may be represented by the rank-one array whose elements are the extents in each dimension. For example:

```
REAL a(5, 5), b(5, 5), c(0:4, 2:6);  LOGICAL lt(5, 5)
...
a = c              !legal since same shape
...
lt = a .b
```

Array operations are element-by-element, not algebraic. For this reason, a statement such as **a = b/c** does not multiply *b* by the matrix inverse of *c* but rather performs the following:

```
a(1,1) = b(1,1) / c(0,2)
a(2,1) = b(2,1) / c(1,2)
a(3,1) = b(3,1) / c(2,2)
...
a(5,5) = b(5,5) / c(4,6)
```

In addition to extending **intrinsic functions** to have array arguments and results, new **intrinsic functions** are also specified for matrix operations. Some of the matrix **intrinsics** are DOT_PRODUCT, MATMUL, SUM (sum of the elements of an array), and TRANSPOSE. Note: the types of the arguments to these functions can be any numeric and, for all but SUM, may be LOGICAL. TRANSPOSE could therefore be of type CHARACTER.

Array constants are introduced. For example:

```
REAL a(5)
a = (/1, 2, 3, 4, 5/)          !set a(1) to 1; a(2) to 2, ...
```

Fortran 90 adds the WHERE statement and construct to qualify array elements for an operation, as in

```
WHERE ( c /= 0 ) a = b/c      !WHERE STATEMENT
```

or

```
WHERE ( c .NE. 0 )                !WHERE CONSTRUCT
   a = b/c
ELSE WHERE
   a = 0
END WHERE
```

NEW CONTROL STRUCTURES

Modern control structures have been added. The new control structures are DO/ENDDO, DO WHILE, CYCLE, EXIT, and CASE. A DO construct is named by *do-construct-name*. This name can be used for documentation, for balancing DO loop closes, and as the target of CYCLE and EXIT statements. These are required to exit from a nest when executing within the innermost DO.

The example in Listing 3 illustrates these features more clearly than words can. A caveat is in order, however: no claim is made that this is the best way to do things, and the source has not been compiled.

MODULES AND INTERFACE BLOCKS

Fortran 90 encourages abstract-data-type programming by introducing the USE statement and the concepts of MODULE, overloaded operator, declared type, and user-defined generic subprograms. These concepts permit MODULEs to be written, standardized, and used throughout the Fortran programming community as libraries are distributed today.

As an example, the Fortran programming community has asked the X3J3 committee for dynamic-length strings. The example of a string MODULE in Listing 4 is greatly condensed from an ISO project worked on by C. Weber and W. Walter and being finished by J. Lawrence Schonfelder. Although this listing is incomplete and sketchy, it illustrates important Fortran 90 features.

Interface blocks specify the calling sequence of two subprograms that may be invoked from the unit in which they appear, resulting in better documentation and diagnostics. An INTERFACE BLOCK for the routines in a graphics library could be maintained in an INCLUDE file. All program units that INCLUDEd that INTERFACE BLOCK would have the library's calling sequences checked for number, class (EXTERNAL, SCALAR, ARRAY, etc.), and type. For example:

```
INTERFACE

   FUNCTION length(record, template)
   INTEGER length;  CHARACTER*(*), INTENT(IN) :: record
   CHARACTER*(*), OPTIONAL :: template
   END FUNCTION length

   SUBROUTINE name(record, template)
   CHARACTER*(*) record, template
   END SUBROUTINE name

END INTERFACE
```

Use of an INTERFACE BLOCK allows user-defined generic subprograms, argument association, and operator overloading.

TABLE 4: Pointer states

UNDEFINED: The initial state, or the state if the pointer's target was never allocated, if it is deallocated other than through the pointer, or if execution of a **RETURN** or **END** statement causes the pointer's target to become undefined.

ASSOCIATED: The state when the pointer is allocated, or associated with a pointer (assigned the information required to reference the values associated with the name), or associated with a scalar or array that has the **TARGET** attribute and is allocated. (See the example in Listing 2.)

DISASSOCIATED: The state when the pointer is nullified (**new** statement), deallocated, or pointer-assigned (**new** statement) with a pointer that is disassociated.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

In a user-defined generic program, the generic name (used by the programmer to invoke the subprogram) is specified immediately after INTERFACE, and a subprogram is specified for each combination of different argument types.

The compiler invokes (or optimizes by in-lining the appropriate intrinsic code) the proper subprogram based on the generic name and argument types. Arguments may be associated by keywords and OPTIONAL arguments. The following are examples based on the above INTERFACEs:

```
CALL name(template=rec_struc, record=input_buf)
i = length(record = input_buf)
```

Here, **rec_struc** and **input_buf** are program-unit CHARACTER variables.

Operator overloading provides for defining operators to maximize the power of MODULEs. When the compiler scans an operator with its typed operands, it uses these definitions to invoke the appropriate subprogram as defined in the MODULE. Fortran 90 uses the INTERFACE BLOCK followed by the appropriate subprogram structure to define operators:

```
INTERFACE ASSIGNMENT( = )  !Subprogram must be 2-arg SUBROUTINE
```

or

```
INTERFACE OPERATOR ( + )   !Subprogram must be 2-arg FUNCTION
```

The keywords ASSIGNMENT(=) and INTERFACE OPERATOR(*defined-operator)* are required. The *defined-operator* is one of the intrinsic operators. It is either unary, binary (+, -, *, /, .NE., ...), or a user-defined operator of the form ".name." where name cannot be either "true" or "false." See the "Modules" section for examples of overloaded operators.

INTERNAL PROCEDURES

Procedures that are local to a program unit have been standardized in Fortran 90. These INTERNAL PROCEDUREs may have their own set of declarations and invoke other internals but can be accessed only from within the host subprogram. The advantage over a global procedure is that variables in the host program unit are available to the internal procedure without placing them in a COMMON or having them serve as the calling sequence arguments. This is similar to the way USE and MODULE eliminate the need for the concept of COMMON. An INTERNAL PROCEDURE may not host an INTERNAL PROCEDURE—i.e., no nesting. A trivial example is in Listing 5.

INPUT/OUTPUT

Fortran 90 adds new OPEN statement specifiers for file positioning, record padding, action (mode), and character delimiters.

Tired of the CONFIG.SYS shuffle?

Let BOOTCON simplify your life.

"... it's bulletproof ... The interface is well thought out. You need this program."
Dr. Jerry Pournelle, *Byte*, July 1990

"Highly recommended."
John C. Dvorak, *PC Magazine*, July 1990

How many CONFIG.SYS and AUTOEXEC.BAT combinations do you have on your PC? BOOTCON eliminates the confusion by storing all your configuration information in just one set of configuration files. Each time you boot your PC you can select the configuration you want to use from a menu of up to 26 choices.

BOOTCON is ideal when you need to use conflicting programs such as Windows 3.0, QEMM/386, 386MAX, Soft-ICE, and Lotus 123 3.0. A default configuration will automatically be used if you do not make a selection within the specified time-out period. BOOTCON also provides an optional password protection feature.

BOOTCON lists for $59.95 and comes on both 5¼" and 3½" disks. It has an unconditional 30-day money-back guarantee. MS-DOS or PC-DOS 3.10 or later is required.

Modular Software Systems

115 West California Blvd., Suite 113 • Pasadena, CA 91105 (818) 794-7602

Windows 3.0 and MS-DOS are trademarks of Microsoft Corporation. PC-DOS is a trademark of International Business Machines Corporation. QEMM/386 is a trademark of Quarterdeck Office Systems. 386MAX is a trademark of Qualitas Corporation. Soft-ICE is a trademark of Nu-Mega Technologies. Lotus and 123 are trademarks of Lotus Development Corporation.

Circle 32 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

File positioning (POSITION=) can be ASIS, REWIND, or APPEND. It supports sequential access only. New files and REWIND files are positioned at the beginning of the file, and APPEND is positioned at the end (the next WRITE destroys an end-of-file). ASIS leaves the file position unchanged if it is already OPENed, and undefined if it is not OPENed. The default condition is ASIS.

Record padding (PAD=) allows values of either YES or NO. A formatted input record is blank-padded when the input list requires more data. A fatal error occurs if PAD=NO and more data are required. The default is YES.

As in C, the action mode for a file operation (ACTION=) can be READ, WRITE, or READWRITE. It limits I/O statements as OPENed. The default in this case is processor-dependent.

Character delimiters (DELIM=) can be APOSTROPHE, QUOTE, or NONE. Character output is delimited with the specified character. The default is NONE.

NAMELIST is now supported for file I/O. An example of a NAMELIST is found in Listing 6.

Fortran 90 adds support for nonadvancing I/O. Fortran has traditionally been a record-oriented I/O system; i.e., at the end of a READ statement, the processor advanced to the end of the record. Fortran 90 changes this with the introduction of the ADVANCE=, SIZE=, and EOR= specifiers for READ statements.

The ADVANCE= specifier requires either YES or NO. If YES, FORTRAN 77 I/O occurs. If NO, the file is positioned at the character where the last I/O finished. To READ variable-length records and to be notified of their lengths, use the SIZE=*integer-scalar* specifier. The specifier is defined with the character count transferred by data-edit descriptors during statement execution. EOR=*label* specifies to continue execution at *label* if an end-of-record condition occurs.

AND THAT'S NOT ALL

In spite of the length of this article, a number of the features of Fortran 90 have not been discussed here. The list of omissions that comes to mind includes ARRAY SECTIONs, VECTOR SUBSCRIPTs, and ARRAY-VALUED FUNCTIONs. There could also have been more on INTRINSIC FUNCTIONs, INTRINSIC SUBROUTINEs, DO, END DO, EXIT, CYCLE, and FORMAT SPECIFIERs (G has been extended, and B, O, Z, EN, and ES have been added). The list of topics not covered, or not fully discussed, is one of the reasons for the two references given below.

Because X3J3 is still working on the standard, think carefully before ordering a copy in the near future. The final version of Fortran 90 should be published by the end of this year. I predict that the standard will be finished some time during this year and that implementations will be appearing before the middle of 1993. PJ

REFERENCES

- *Fortran 90*, X3J3 Standing Document 8 from meeting 115, June 1990.
- *Fortran Forum*, a SIGPLAN Quarterly Publication on Fortran published by ACM.


```
      CHARACTER, DIMENSION(:), POINTER  ::  chars
    END TYPE string

    INTERFACE ASSIGNMENT(=)
      MODULE PROCEDURE    ! MODULE SUBROUTINEs defined below
    &  s_eqs_s,           ! CALLed if string = string parsed
    &  s_eqs_c,           ! CALLed if string = char parsed
    &  c_eqs_s            ! CALLed if char = string parsed
    END INTERFACE

    INTERFACE OPERATOR(//)
      MODULE PROCEDURE    ! MODULE FUNCTIONs defined later
    &  s_concat_s,        ! Invoked if string // string parsed
    &  s_concat_c,        ! Invoked if string // char parsed
    &  c_concat_s         ! Invoked if char // string parsed
    END INTERFACE

!  Note: The ISO_MODULE defines relational operators
!  Note: The ISO_MODULE defines its INTRINSIC FUNCTIONs LEN, INDEX
!  Note: The ISO_MODULE defines type conversions for internal use
!        ...  A lot more code!
    SUBROUTINE s_eqs_s(st, ss)! Compiler CALLs when
                              ! string = string is parsed
    TYPE (string) INTENT(OUT)  ::  st
    TYPE (string) INTENT(IN)   ::  ss
    IF ( .NOT. ASSOCIATED(ss%chars) ) CALL error
    IF ( ASSOCIATED(st%chars) ) THEN
      IF ( ASSOCIATED(ss%chars, st%chars)  ) RETURN
      NULLIFY (st%chars)
    ENDIF
    st%chars = ss%chars
    END SUBROUTINE s_eqs_s
!       ... A lot more code!
    END MODULE ISO_string

!  Using the string MODULE
    USE string                ! The MODULE
    TYPE (string) s1, s2      ! MODULE has type definition
    ...
    s1 = 'abc def '           ! Trailing blank preserved, s_eqs_c
    s2 = 'ghi jkl mno'
    ...
    s1 = s2 // s1             ! // is overloaded operator, compiler
                              ! invokes function s_cat_s(s2,s1) then
                              ! CALL s_eqs_s(s1,string_temp)
    PRINT *, s1               ! Compiler prints structure components
    END
```

LISTING 5

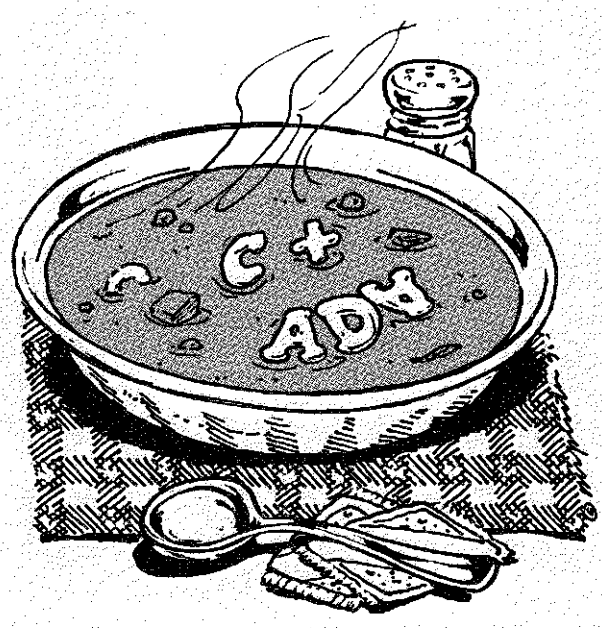
```
    SUBROUTINE sub
    CALL s
    PRINT *, i, j             ! i & j are known to internals

    CONTAINS                  ! Required, separates host & internals

    SUBROUTINE s
    i = nj(5);    END         ! i not declared locally, must be host
    FUNCTION nj(k)
    j = k+5;      END         ! j not declared locally, must be host

    END SUBROUTINE sub
```

LISTING 6

```
    NAMELIST /study_params/ temp, pres, volume
    REAL, PARAMETER  ::  n = 6.02252E23,  R = 0.0823
10  PRINT *, 'To terminate, enter both values as 0'
    PRINT *, 'If not changing both params end with /, no ,'
    PRINT *, 'Input: &study_params temp=value, pres=value/'
    IF ( temp .EQ. 0  .AND. pres .EQ. 0 ) STOP 'All done'
    READ  (*, NML=study_params)
    volume = n*R*temp/pres        ! Remember:  PV = nRT
    WRITE (*, NML=study_params)   ! Outputs: temp, pres, & volume
    GO TO 10
    END
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Pragma interface



Combining Ada and C routines in large systems

By Richard Riehle

When Ada was being defined as a programming language, the design team recognized the need for the occasional leveraging of source code from other languages. The reality is that it takes time to develop programs in a new language, and often a large body of usable code is available in some other language. For example, much of the weapons systems software from 1965 through the present was developed using Fortran and assembly language. The mathematics routines that exist in Fortran are relatively well designed, and engineers rightfully have some confidence in them. Many of these routines are not yet coded in Ada—an Ada programmer can incorporate them into a design without recoding them.

Since C's advent in the early 1970s, a large body of code has been developed in it. In addition, environments such as X Window using the XLib interface are designed around calls to predefined C functions. Rather than rewrite these functions entirely in Ada, it makes sense to take advantage of Ada's ability to interface easily with other languages.

Before examining specific examples of Ada interfaces to other languages, it is important to note that such language inter-operability may violate some important principles of Ada. For example, it can be difficult to enforce Ada's strong typing model when interfacing to a weak-typed language such as C. More importantly, the system designer would be wise not to use this feature of Ada to "work around" the requirement of developing Ada source code.

Richard Riehle is a software engineer with AdaWorks. He can be reached at 1090-105 Tanland Drive, Palo Alto, CA 94303; (405) 858-1551.

SOME PROS AND CONS OF C

Languages such as C++ and Ada support some important principles of software engineering: information hiding, levels of abstraction, modularity, and localization. Because C does not support these principles very well, it is generally a poor choice for large, important software engineering projects. Large programs in C tend to be more difficult to maintain than programs with strong support for levels of abstraction. However, it is often valuable to write small routines in C when the software requires a device driver, access to some system-level or graphics facility, or an improvement in execution speed.

Using Ada, a large system can be designed that has most of the source code in Ada but that also has selected parts in C, Fortran, or even assembly language. My own choice is to code those selected parts in C, which serves as a kind of universal assembler because of its portability and efficiency.

A pragma is a compiler directive, and Ada has quite a few of them. The *Ada Language Reference Manual* (*ALRM*) explicitly defines a special *interface* pragma in this way:

Takes a language name and a subprogram name as arguments. This pragma is allowed at the place of a declarative item, and must apply in this case to a subprogram declared by an earlier declarative item of the declarative part or package specification. This pragma is also allowed for a library unit; in this case the pragma must appear after the subprogram declaration, and before any subsequent compilation unit. This pragma specifies the other language (and thereby the calling conventions) and informs the compiler that an object module will be supplied for the corresponding subprogram.

The *ALRM* syntax for this pragma is as follows:

```
pragma INTERFACE (language_name,subprogram_name);
```

In theory, a pragma interface could be available for any language for which there are object modules. What happens in practice, however, is that the Ada compiler vendors limit support to languages such as assembly, Fortran, and C. A wider range of languages will be supported in the future. Some compiler vendors support less well known languages, such as JOVIAL.

Suppose you needed to use a Fortran library of trigonometric functions as part

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

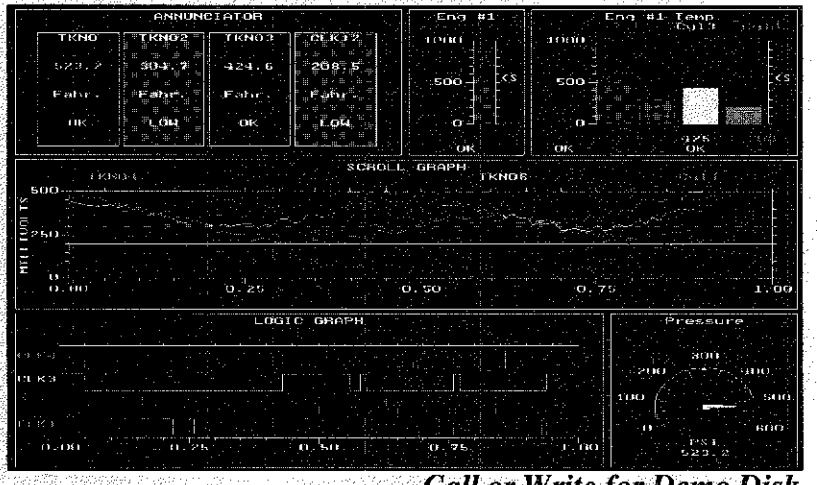
of an Ada implementation. The code might look like this:

```
package TRIG_LIB is
  function SIN (X : Float) return FLOAT;
  function TAN (X : Float) return FLOAT;
private
  pragma Interface (FORTRAN, SIN);
  pragma Interface (FORTRAN, TAN);
end TRIG_LIB;
```

A simple program to use this package is shown in Listing 1. The pragma interface code is placed in the private part of the package specification in order to "hide" it from the client procedure, **Use_Trig**. Listing 2 demonstrates interfacing Microsoft C, using the Meridian AdaVantage compiler. The code also works with the new Open Ada compiler from Meridian. The MSCTEST example in Listing 2 has several procedures for which we have declared a pragma interface. Meridian and Alsys Ada provide a predefined capability for interfacing to object modules produced by the Microsoft C compiler. The parameter list of the C program must correspond to that of the Ada subprogram. It is important that the object module be in a format recognized by the Ada compiler. If it isn't, setting up a mechanism to make the pragma work involves additional low-level tinkering.

Real-Time Graphics and Measurement/Control Tools

Programmers Tools for Microsoft C, Turbo C, C++ and Turbo Pascal



Call or Write for Demo Disk

Fast Real-Time Graphics

Real-time graphics routines for scrolling graphs, sweep graphs, process control bargraphs, annunciator panels, 3 types of meters and general graphics text displays. Real-time mouse routines are also included. The routines are optimized around the graphics primitives which come with the respective compiler.

Powerful Measurement and Control

PID Control (position and velocity algorithms), Thermocouple linearization for the 15 standard TC types (B, BP, BN, E, J, JP, JN, R, K, KP, KN, S, T, TP, TN), Thermocouple curvefitting for any temperature range, Fourier analysis routines including 16K point FFTs.

Source Code and Royalty Free

No additional charge for source code, use these routines in your programs without worrying about royalty fees. No other third party drivers are necessary.

Each version is $200

Please specify compiler when ordering.
Price includes libraries, source code and a 250 page manual.
Mastercard, Visa accepted. Shipping charge $5 within USA, $9 Canada and $25 elsewhere.

Quinn-Curtis, 21 Highland Circle, Needham, MA 02194 USA Tel. 617/449-6155
FAX 617/449-6109

Circle 33 on Reader Service Card

THREE PRAGMA PARAMETERS

The parameter list of the pragma interface directive, as defined in the *ALRM*, contains **language** (the source language used to produce the object module) and **subprogram** (the source name of the object module; this is usually the name used for the Ada subprogram as well).

For many compilers, including the Meridian compiler being used for the examples in this article, there is a third parameter: **link name**. When the Ada subprogram name differs from the object symbol name, that name can be supplied with a separate parameter. This is useful for Ada programmers who want to use more-meaningful procedure names for users of the specification.

The subprogram for the MSCTEST procedure has the same names as the object modules produced by the Microsoft C compiler. Two of the Microsoft C programs are shown in Listing 3.

A close examination of the parameter list for the function **cinttst1** shows that the *in* parameter is an integer *x*. The return data type is also an integer. Note that the types for parameters must be compatible. I know of no compiler that does automatic type conversion under pragma interface; the programmer must preserve type continuity.

With floating point interfaces, type continuity becomes a tricky issue. For Meridian it is good to design floating point interfaces as Ada access types because Microsoft C returns a *float* (or *double*) result as a pointer. Also, the Meridian Ada compiler supports only 64-bit (*double*) *float* types, so variables of this type can be used as *in* parameters to a function. You might want to set up the interface as shown in Listing 4.

A procedure to use this package could be a variation of the earlier version of **Use_Trig** in this article. See if you can spot the difference between the version in Listing 5 and the earlier one.

When using pragma interface, remember the underlying data representations. Designers of Ada package

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

specifications should raise the level of abstraction as much as possible so that such knowledge is not necessary to its users. This is why the pragma interface directive is put in the private part of the package. In addition, the pragma must appear in the same compilation unit of the subprogram as that named in its parameter list.

One could consider raising the level of abstraction even more by further encapsulating the **Trig_Lib** routines. An example is given in Listing 6.

With this abstraction, the user of the package need not know about access types or be confused by reading the private part of the specification, and can use the package just as it reads to do the appropriate trigonometric functions. This use of information hiding minimizes confusion by managing the complexity within the package body.

Notice that whenever we use a pragma interface, no other code is provided for the Ada subprogram that is being interfaced. It would be illegal to include such code; all of the code must be contained in the called program.

PLATFORM SPECIFICITY

One key problem with pragma interface is the issue of platform specificity. The Ada system should be designed at the highest level of abstraction; on the other hand, any pragma interface code should be at the lowest level possible. This is especially true if you are interfacing to assembly language but also holds true for C interfaces. By keeping the interface pragmas at the lowest level of abstraction, you ensure the portability of your Ada code and, consequently, of your system design. Ada is more portable than assembly code and usually more portable than C. By coding your C routines at the lowest level of abstraction, you enhance the maintainability of your system.

Many Department of Defense contracts prohibit the use of pragma interface to languages such as C. However, given the ever-increasing use of the Ada language in the commercial market, it makes good sense to use the right tool for the job. On occasion, C is the right tool. One Ada compiler executive even admitted to me that five percent of his company's Ada compiler is written in the C language.

Pragma interface for C is supported in MS-DOS environments by Alsys and Meridian. Nearly every Ada compiler vendor has support for interface to some level of assembly language.

You can expect all Ada compilers to have a few "gotchas"; these need to be scrupulously avoided when using pragma interface. Be sure to read the user manual carefully. Packages that require pragma interface should always hide the pragma code within the package body. The fact that this hiding requires a little more source code within the body simplifies the package specification substantially. PJ

Tired of the limitations of MS-DOS?
Thinking about switching to another OS?
Well DON'T, now there is:

PRO-CLI v4.1

CLI is a tiny, intelligent front & back end processor that enhances the MS-DOS system kernel. CLI provides many features of advanced OS's, like VMS & UNIX, to MS-DOS without the tremendous costs and additional training.

CLI-Types: ALIASs, LOGICALs, SYMBOLs, LEXICAL-FUNCTIONs and STREAMs are available to the command line, batch programs, & all existing application programs. CLI is 100% MS-DOS compatible, and NO changes are required in any software to reference any CLI-Types.

Some of the features offered by PRO-CLI include:

- batch programs with numeric & string processing
- facilities for source code and project management
- transparent task-to-task communications
- file & device management from the command line
- command line editing, history, and recall
- flexible system installations & re-configurations
- user configurable PRO-CLI resources

For the first time, designers, developers, and managers can coordinate software development & production efforts without the major restrictions of MS-DOS.

Call or Write for Additional Information!

Trademarks: VMS of D.E.C. MS-DOS of Microsoft UNIX of AT&T

Alphax Systems, Inc.
1005 Sussex Blvd.
Broomall, PA 19008
(215) 543-2658

Price: $ 149
S/H: $ 6

Major Credit Cards Accepted

Circle 34 on Reader Service Card

Maximum Software Security

100% Surface Mount

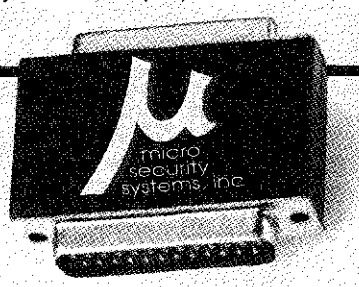
New-Macintosh Version

SecuriKey is a hardware device designed to prevent unauthorized use and duplication of computer software.

- Security leader with 8 different devices
- Works on all IBM/compatibles and Macs
- Transparent to printer
- Allows unlimited back-up copies
- Protects domestic & intern'l sales profits

Call Today! (800) 456-2587

Micro Security Systems Inc. 4750 Wiley Post Way, Suite 180
Salt Lake City, Utah 84116 (801) 575-6600 FAX (801) 575-6621



Circle 35 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LISTING 1

```
with Text_IO, Trig_Lib;
procedure Use_Trig is
    Parm_Value : Float := 0.0;
    Retn_Value : Float := 0.0;
    package FLT_IO is new Text_IO.Float_IO(Float);
begin
    Text_IO.Get(Parm_Value);
    Flt_IO.Get(Parm_Value);
    Retn_Value := Trig_Lib.Sin(Parm_Lib);
    Text_IO.Put("The SIN is ");
    Flt_IO.Put(Retn_Value);
    Text_IO.New_line;
    Retn_Value := Trig_Lib.Tan(Parm_Lib);
    Text_IO.PUT("The Tangent is ");
    Flt_IO.Put(Retn_Value);
end Use_Trig;
```

LISTING 2

```
with Text_IO;   use Text_IO;
    package FIO is new Text_IO.Float_IO(Float);
    package IIO is new Text_IO.Float_IO(Integer);

procedure MSCTEST is        -- MSC = Microsoft C
    i, j: integer;          -- compatible with Microsoft C
    a, b: float;            -- Meridian is 64 bits; be careful

    procedure msc_start;  -- procedure specification
        pragma INTERFACE(microsoft_C, msc_start);
                        -- This pragma replaces the body for msc_start.
                        -- No other code is permitted for msc_start
                        -- once it is implemented via a pragma interface
    procedure msc_end;
        pragma INTERFACE(microsoft_C, msc_end);

    function cinttst1(x: integer) return integer;
        pragma INTERFACE(microsoft_C, cinttst1);
    procedure cinttst2(x, y: integer; z: out integer);
        pragma INTERFACE(microsoft_C, cinttst2);
    procedure cflttst(x: in out float; y: in out float);
        pragma INTERFACE(microsoft_C, cflttst);

begin                       -- MSCTEST
    Text_IO.Put_Line("Beginning AdaVantage/Microsoft C Interface Example.");
    Text_IO.NEW_LINE;

    TEXT_IO.Put_Line("Calling msc_start.");
    MSC_START;

    TEXT_IO.Put_Line("Calling cinttst2 ...");
    TEXT_IO.New_Line;
    I := 5;
    cinttst2(I, 11, J);
    TEXT_IO.New_Line;
    TEXT_IO.Put_Line("... result returned from cinttst2:");
    TEXT_IO.Put("  J (55) = ");
    Iio.Put(j);
    TEXT_IO.New_Line(2);

    TEXT_IO.Put_Line("Calling cflttst ...");
    TEXT_IO.New_Line;
    A := 1.23456;
    B := 9.87654;
    cflttst(A,B);
    Text_IO.New_Line;
    Text_IO.Put_Line("... results returned from cflttst:");
```

continued

People are talking about us.

F77L-EM/32 & Lahey Ergo OS/386

Port mainframe programs as large as 96MB to 386/486's with the 32-bit DOS-Extender compiler. The Winner of *PC Magazine's* 1988 Technical Excellence Award just got better. New Version 4.0 includes: Programming Tools, Popular Fortran 90 features, Virtual Memory Support, DESQview Support, New Documentation and Free Unlimited Runtime Licenses. F77L-EM/32 $995 OS/386 $395

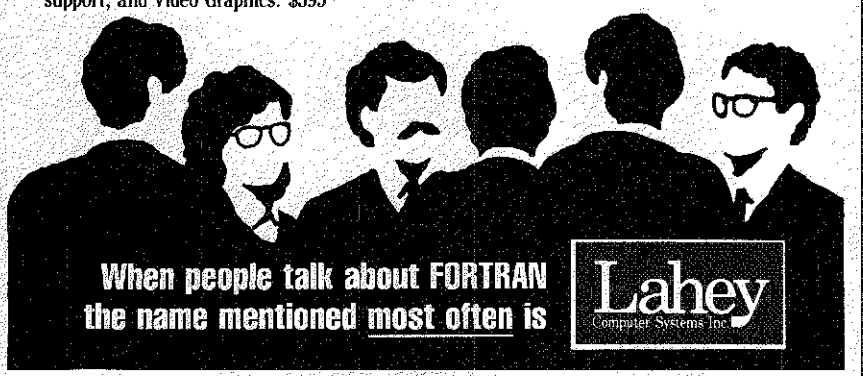
NEW RELEASE

F77L

The fastest real-mode compiler available. F77L can take advantage of your 386 PC by generating 32-bit instructions. New Version 5.0 includes: Fortran 90 features, Weitek support, and Video Graphics. $595

Lahey Personal Fortran 77

Version 3.0: Full ANSI 77, Editor, Debugger, Linker, Library Manager, Microsoft and Borland C interfaces. A great learning tool at an unbeatable price. $99



Contact us to discuss our products and your needs. **(800) 548-4778**
Lahey Computer Systems, Inc. P.O. Box 6091, Incline Village, NV 89450
Tel: (702) 831-2500 FAX: (702) 831-8123

FORTRAN IS OUR FORTE

Circle **36** on Reader Service Card

Complete Modem Communications

TURBO PASCAL & C programmers

Introducing
Sugar Coated Serial™

A complete library of Turbo Pascal & C procedures, functions and low-level assembly routines that allows you to EASILY incorporate MODEM communications into your application.

Features:

- All source code included (including Assembler).
- No knowledge of assembler required.
- No royalties or fees for use in your programs.
- Easily do XMODEM, YMODEM, & Telink file transfers (including multiple files with YMODEM.)
- Pull-down menu communications program included using only routines from this library.
- Complete user manual including many examples.
- Much more...call for the information pak to get the full details.

For IBM PC/XT/AT compatibles using T.P v5.0 or >

* Complete System: $109.95
* Demo Disk: $ 5.95
* Information Pak: Free

Shipping included, IL residents add 6.25% sales tax. Specify disk size.

Send Check, M.O. or Credit Card info with order to:

Goin Places Software
Suite 145
1202 N West 75th Street
Downers Grove IL 60516

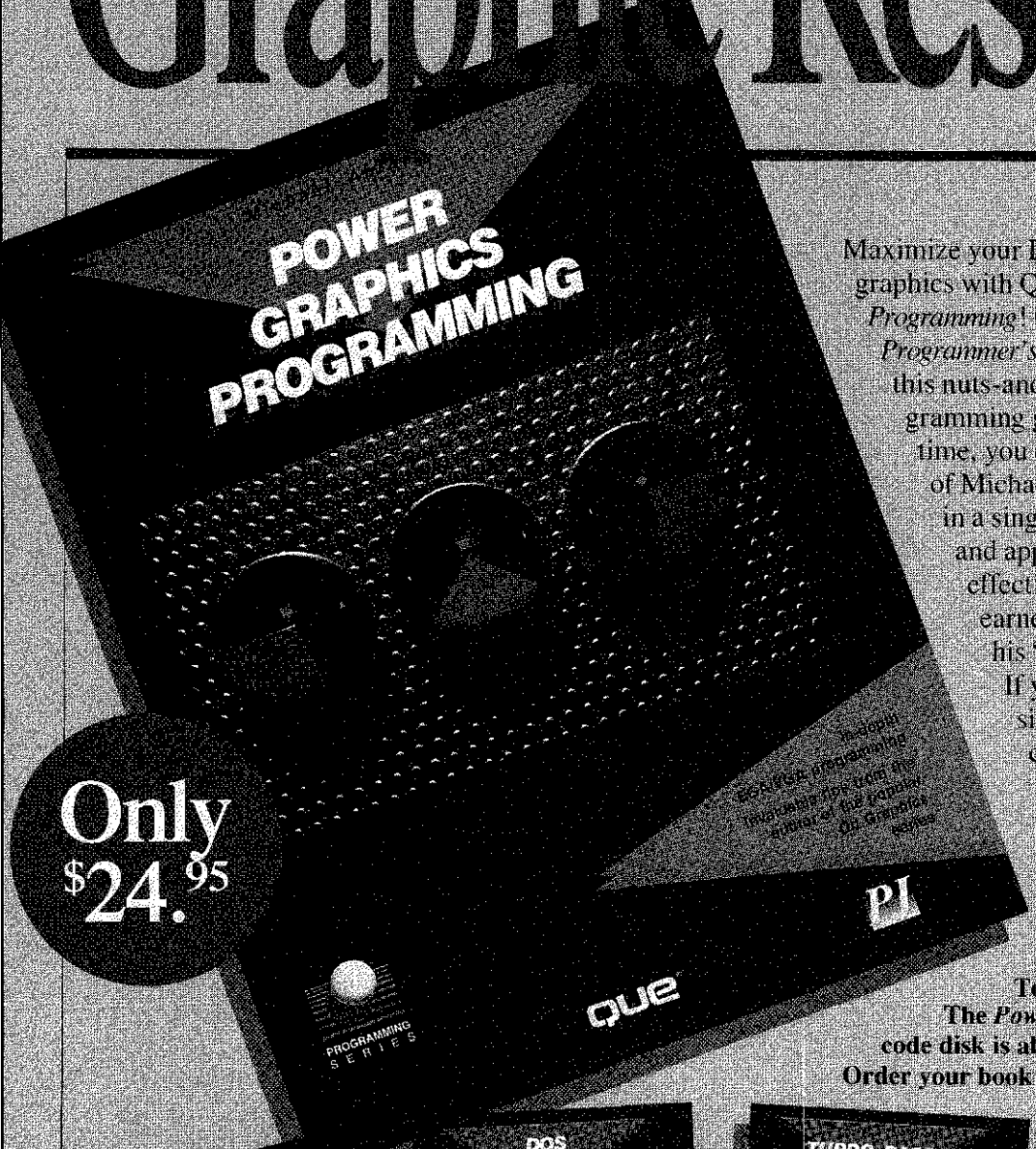
For phone orders or info call: **(708) 759-1270**

Circle **37** on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The complete *Programmer's Journal* "On Graphics" series—assembled in one volume for the very first time!

Graphic Results!

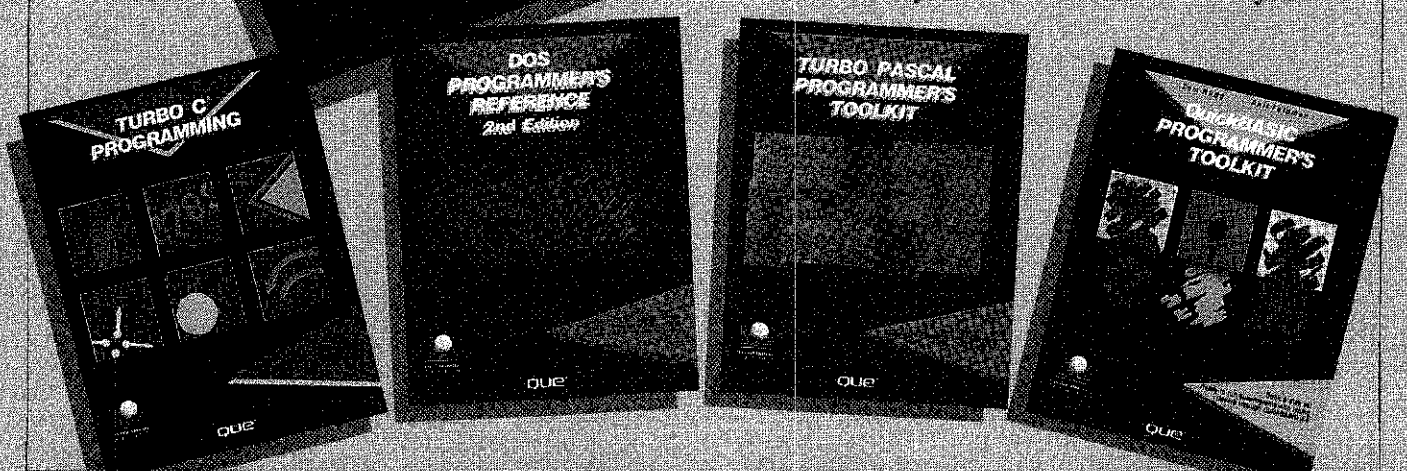


Only $24.95

Maximize your EGA/VGA programming graphics with Que's *Power Graphics Programming*! Que has teamed up with *Programmer's Journal* to bring you this nuts-and-bolts graphics programming guide. Now, for the first time, you can have the expertise of Michael Abrash consolidated in a single volume. His insights and approach to efficient, effective programming have earned him high acclaim for his "On Graphics" series. If you are seeking that elusive graphic edge, you can't be without *Power Graphics Programming*!

Order yours directly from *Programmer's Journal*

To order call 800-234-0386. The *Power Graphics Programming* code disk is also available for $14.95. Order your book and code disk today!



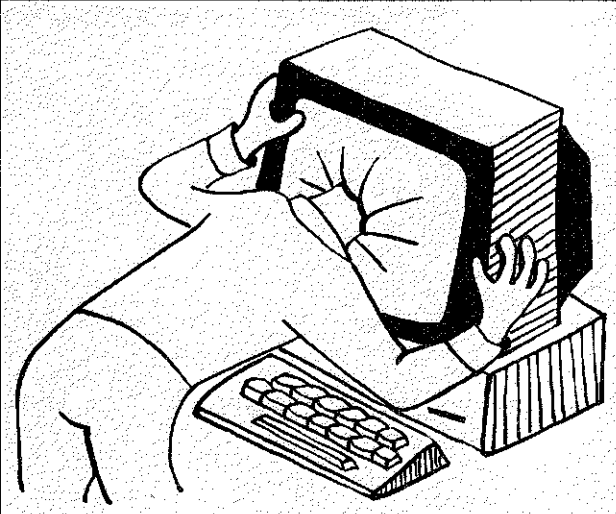
Look for these other practical Programming books from Que

Available in better bookstores and computer stores everywhere. Or call 800-428-5331, ext. PJ01.

que®

PRODUCED BY 2004 BY MJZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Contact this advertiser directly



Look inside your software!

Quaid Analyzer shows the inner workings of software, even without the source. Scroll through text or instructions as easily as with an editor.

You can get a report on the screen at every DOS service call, or every interrupt or breakpoint.

Ordinarily **Quaid Analyzer** is memory resident. For demanding tasks, load it before DOS, then watch DOS load and install device drivers. **Quaid Analyzer** includes ways of tracing in spite of countermeasures used to make tracing difficult. We use it to unravel copy protection. **Quaid Analyzer** is pure software, so you don't have to take apart your computer.

To order **Quaid Analyzer**, call us with your credit card, or send us a check for $200 US funds. We ship within a day at our expense.

Quaid Analyzer comes with a manual, and software on a 3 inch and a 5 inch diskette. It is not copy protected. If you are not satisfied, you can return it within 30 days for a refund.

Quaid Software Limited

45 Charles Street East, Third Floor Dept P
Toronto Ontario Canada M4Y 1S2

(416) 961-8243 fax: (416) 961-6448

Contact this advertiser directly


```
    Text_IO.Put("  a (8.76543) = ");
    FIO.Put(A, aft => 5, exp => 0);
    Text_IO.New_Line;
    Text_IO.Put("  b (2.34567) = ");
    FIO.Put(b, aft => 5, exp => 0);
    Text_IO.New_Line(2);

    Text_IO.Put_Line("Calling msc_end.");
    MSC_END;

    Text_IO.New_Line;
    Text_IO.Put_Line("End of Ada)C Interface Example.");
end MSCTEST;
```

LISTING 3

```
void cflttst( double far *x,double far *y) {
    printf("> Beginning cflttst.\n>    Values passed from msctest:\n");
    printf(">      x (1.23456) = %g\n", *x);
    printf(">      y (9.87654) = %g\n", *y);
    *x = 8.76543;
    *y = 2.34567;
    printf("> End of cflttst.\n");
    }
void cinttst2( int x, int y, int *z) {
    printf("> Beginning cinttst2.\n>   Values passed from msctest:\n");
    printf(">      x ( 5) = %d\n", x);
    printf(">      y (11) = %d\n", y);
    printf("> End of cinttst2.\n");
    *z = x * y;
}
```

LISTING 4

```
package Trig_Lib is
    type Pointer is access Float;
    function Sin (X : Float) return Pointer;
    function Tan (X : Float) return Pointer;
private
    pragma Interface (Microsoft_C, sin);
    pragma Interface (Microsoft_C, tan);
end Trig_Lib;
```

LISTING 5

```
        with Text_IO, Trig_Lib;
procedure Use_Trig is
    Parm_Value : Float := 0.0;
    Retn_Value : Float := 0.0;
    package FLT_IO is new Text_IO.Float_IO(Float);
begin
    Text_IO.Get(Parm_Value);
    Flt_IO.Get(Parm_Value);
    Retn_Value := Trig_Lib.Sin(Parm_Lib).all;
    Text_IO.Put("The SIN is ");
    Flt_IO.Put(Retn_Value);
    Text_IO.New_line;
    Retn_Value := Trig_Lib.Tan(Parm_Lib).all;
    Text_IO.PUT("The Tangent is ");
    Flt_IO.Put(Retn_Value);
end Use_Trig;
```

LISTING 6

```
package Trig_Lib is
    function Sin_Of (X : Float) return Float;
    function Tan_Of (X : Float) return Float;
end Trig_Lib;

package body Trig_Lib is
    type Pointer is access Float;
    function Sin (X : Float) return Pointer;
        pragma Interface (Microsoft_C, sin);
    function Tan (X : Float) return Pointer;
        pragma Interface (Microsoft_C, tan);

    function Sin_Of(X : Float) return Float is
    begin
        return Sin(X).all;
    end Sin_Of;

    function Tan_Of(X : Float) return Float is
    begin
        return Tan(X).all;
    end Sin_Of;
end Trig_Lib;
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

String handling

Assembly language functions for manipulating BASIC strings

By Murray L. Lesser

Compiled BASIC enhanced with assembled procedures is one of the best languages for data-processing applications involving complex string handling. Unfortunately, prior to BASIC 7.0 there was no documented way for programmers using compiled BASIC to write an assembly language function returning a BASIC string. According to the *Microsoft MASM Mixed Language Programming Guide*, it was catastrophic to contaminate string space by creating a new BASIC string from an assembly language procedure or even by assigning a new value to an existing string that would change its length.

The restriction prohibiting string functions in assembly language was officially lifted in version 7.0 by the addition of four "mixed-language" procedures to the BASIC link libraries. Two functions substitute for the traditional method of locating the representation of a string in string space, and two subroutines allow creating, or modifying the length of, BASIC variable-length strings from the fixed-length strings used in assembly language. But what the version 7.0 documentation doesn't mention is that these "new" library calls are pseudonyms for four standard BASIC link library procedures that have long been available in Microsoft BASIC compilers—at least since BC version 4.0.

This article discusses these four library procedures and illustrates them with a function that returns a string containing the volume label (if any) of the disk specified in the function-call argument. The routine links only to BASIC programs compiled with BC version 7.0 or later, but enough information is presented to let programmers edit the source code for linking to programs compiled with BC versions 4.x and (presumably) 6.0.

STRINGLENGTH AND ADDRESS

Two newly named library functions let assembly language programmers locate the representation of any string variable in any string space. These functions became necessary with the BASIC 7.0 enhancement that allowed optional compilation for multiple "far string" spaces, thereby giving programmers increased data and string space at the expense of performance. Although the technique of obtaining the current length and location of a near string from the four-byte string descriptor is still valid, Microsoft has chosen to keep the format of the multistage far-string descriptor proprietary. All that is publicly known is that the first descriptor stage for a far string variable is a double word in DGROUP.

StringLength returns the length of the string as an integer in AX; **StringAddress** returns a far pointer in DX:AX to the actual string in whatever string space it may occupy. Both functions take as their argument the "value" of the near pointer to the string descriptor in DGROUP. (Fortunately, this value is the "reference" passed by any calling BASIC routine that uses the string variable as an argument.) These two library functions can be used when copying the contents of all or part of any BASIC string to an assembly-language fixed string of sufficient length, as shown in VLABEL.ASM (Listing 1). Because the link libraries for near- and far-string compilations are separate, an assembled procedure using these function calls will link properly to programs compiled for either string mode.

Obviously, these two string locating functions are not needed when writing procedures to be linked to BASIC programs compiled with versions of BC prior to 7.0; during that time only near strings were supported. Even so, the functions were available under their original names. **StringLength** is a new public name for the old link-library function **B$FLEN**, used by all versions of the compiler to evaluate the **LEN** function. **StringAddress** is also known as **B$SADD**, used by earlier versions of

---

Murray L. Lesser is a retired IBM engineer. He has written three books about enhancing various Microsoft BASIC compilers with assembled modules. The most recent, Advanced QuickBASIC 4.0, *was published by Bantam.*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LISTING 1

```
        PAGE    ,105
        TITLE   VLABEL Function
        .MODEL  MEDIUM
                                VLABEL.ASM
        ;   VLABEL (<drive>) [DECLARE FUNCTION VLABEL$ (A$)]  is an
        ; assembled string function to be linked to programs written in
        ; Microsoft BASIC v. 7.0 or later. When given a valid drive code as
        ; an argument, it returns a string containing the volume label.

        ;   VLABEL uses the first character of the string argument as the
        ; code for the requested drive. If the disk in that drive has no
        ; label (or no such drive exists), VLABEL will return a null
        ; string.  If the code refers to a diskette drive with the door
        ; open, the calling BASIC program will be interrupted with a BASIC
        ;  "Disk not ready" error  interrupt.

        ;  By M. L. Lesser, 4/27/90. Assembled with MASM v. 5.1

        EXTRN StringAddress: FAR
        EXTRN StringAssign:  FAR
        EXTRN StringLength:  FAR
        EXTRN StringRelease: FAR

.DATA
FCB     DB      0FFH                    ;Use extended FCB format to
        DB      5 DUP(0)                ;   avoid unwanted period.
        DB      8                       ;Label attribute
DRIVE   DB      0                       ;Drive code goes here
FLABEL  DB      11 DUP(?)               ;Dummy file name
        DB      21 DUP(?)               ;Remaining FCB filler
DESCRIP DD      0                       ;BASIC string descriptor

.CODE
        PUBLIC VLABEL
VLABEL  PROC
        PUSH    BP
        MOV     BP,SP
        PUSH    DS
        PUSH    DI
        MOV     AH,2FH                  ;Get and save BASIC's DTA
        INT     21H
        PUSH    BX                      ;DTA offset
        PUSH    ES                      ;DTA segment
;  Check for non-null input string (drive code):
        PUSH    6[BP]                   ;Input string descriptor pointer
        CALL    StringLength            ;Get length
        OR      AX,AX                   ;Is it zero?
        JZ      NODRV                   ;If so, skip getting drive code
;  Locate address of input string:
        PUSH    6[BP]
        CALL    StringAddress
        MOV     DS,DX                   ;Move string address to DS:BX
        MOV     BX,AX
        MOV     AL,[BX]                 ;  and drive letter to AL
        AND     AL,5FH                  ;Make drive character upper case
        SUB     AL,40H                  ;Convert to FCB drive code
;  Set up extended FCB to find label:
NODRV:  MOV     BX,SS                   ;Restore DGROUP addressability
        MOV     DS,BX                   ;  to DS and ES
        MOV     ES,BX
        MOV     DRIVE,AL                ;Load drive number in FCB:
        LEA     DI,FLABEL               ;Load FLABEL portion of FCB with
        MOV     AL,'?'                  ;  question marks
        MOV     CX,11
        REP     STOSB
;  Find the first file containing a label (there is only one!):
        LEA     DX,FCB                  ;Set DTA to FCB
        MOV     AH,1AH
        INT     21H
        MOV     AH,11H                  ;Find first
        INT     21H
        OR      AL,AL                   ;Will be zero if label found
        JNZ     NULLIT                  ;If not, send zero-length string
; Decrease CX by count of trailing blanks in label:
        MOV     CX,11
        MOV     AL,' '
        DEC     DI                      ;Restore DI to end of label
        STD                             ;Scan "down"
        REPZ    SCASB
        INC     CX                      ;Include "last" non-blank
        CLD                             ;Restore cleared direction flag
; Convert label to BASIC string
        PUSH    DS                      ;Far address of MASM string
        LEA     AX,FLABEL
        PUSH    AX
        PUSH    CX                      ;Length of string
        PUSH    DS                      ;Far address of descriptor
        LEA     AX,DESCRIP
        PUSH    AX
        XOR     AX,AX                   ;Zero-length copies MASM string
        PUSH    AX                      ;  into BASIC string space
        CALL    StringAssign
DIVOTS: POP     DS                      ;Restore BASIC's DTA
        POP     DX
        MOV     AH,1AH
        INT     21H
        POP     DI                      ;Restore registers
        POP     DS
        POP     BP
        LEA     AX,DESCRIP              ;Return near address to BASIC
        RET     2
NULLIT: LEA     AX,DESCRIP              ;Make BASIC string null
        PUSH    AX
        CALL    StringRelease
        JMP     DIVOTS
VLABEL  ENDP
        END
```


Multitasking

Real-Time

Do the multitasking systems you are considering using seem a bit like overkill or need an awful lot of memory? Maybe all you really need is a system which can run more than one DOS program concurrently. If this sounds like your situation, consider:

MultiDos Plus™

Flexible preemptive scheduling of multiple tasks with full access to MS-DOS or PC-DOS. Supports the loading and execution of tasks in LIM 4.0 expanded memory. For more information please call for a free brochure.

Call our BBS for free demo software.

The complete software package is only $99 + S/H. Visa/MasterCard accepted.

30-day money back guarantee.

(subject to a $15 restocking fee)

Nanosoft Inc.

13 Westfield Road
Natick, MA 01760

800-678-2141

508-651-0091
FAX 508-655-8860
BBS 508-650-9552

All code in this issue is available through our bulletin board services or directly from *PJ* on disk. For information see pages 80 and 81.

Circle 38 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

THIS IS YOUR OPPORTUNITY TO STAY AHEAD OF TODAY'S COMPETITIVE SOFTWARE DEVELOPMENT MARKET.
The conference is created especially for software programmers, developers and technical managers from around the world who know the power and productivity gains realized by using Borland's Turbo Languages. The focus will be on developing applications for windowing environments, increasing programmer productivity and object-oriented programming.

LEARN FROM THE EXPERTS . . .
The conference will highlight more than 55 technical and educational sessions. Presentations will be given by top industry notables from throughout the programming community.

TRACK #1: LANGUAGES AND DEVELOPMENT
Explores applications development using Borland's programming languages. Learn the complexities of building applications with graphical user interfaces.

TRACK #2: TOOLS AND LIBRARIES
Focuses on using programming tools for debugging, profiling, quality assurance and using libraries to increase productivity and program reliability.

TRACK #3: PROJECT MANAGEMENT AND DESIGN
Explores techniques that enable developers to deliver robust products on schedule. Technical managers will be provided with insight on the complex process of managing the development process and building software to specification.

TRACK #4: ELSE
Covers topics including tips, techniques and technologies for getting the most out of your language, OOP frontiers, and high-level application case studies.

These seminars are designed to provide the inspiration, skills and information you need to take a giant step into the next generation of highly productive, software programming tools!

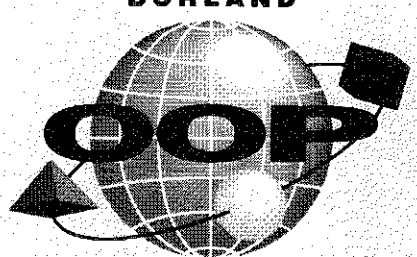
Registration fee—$795

- **"BIRDS OF A FEATHER" SESSIONS**
- **THE LAB** *(an around-the-clock resource center for those who can't wait to apply presented programming techniques)*
- **EXHIBITS** *(talk with over 30 vendors offering products and information)*
- **OPTIONAL PRE-CONFERENCE SEMINAR SESSIONS, SPECIAL GAME ROOM AND THEME MOVIE ROOM.**

If you're interested in attending or exhibiting, please contact:

CT MEETING PLANNERS, INC.
731 MAIN STREET, SUITE C-3
MONROE, CT 06468
1-800-942-TURBO
OR (203) 452-5388

BORLAND



LANGUAGES

BORLAND LANGUAGES CONFERENCE

APRIL 28–MAY 1, 1991
SHERATON PALACE HOTEL, SAN FRANCISCO, CALIFORNIA

"OPENING WINDOWS ON THE NEXT GENERATION OF SOFTWARE"

Copyright © 1991 Borland
BOR 2182

Contact this advertiser directly.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

the compiler to evaluate the **SADD** function and by BC 7.0 to evaluate the **SADD**, **SSEG**, and **SSEGADD** functions. (For BC versions prior to 7.0, **B$SADD** returns only a near pointer in AX.)

STRINGASSIGN AND RELEASE

The major newly documented library string subroutine is **StringAssign** (aka **B$ASSN**). Depending on the values of the arguments used when calling **StringAssign**, it can assign a new value to a string variable from a fixed-length string or from another variable-length string, assign the value of a variable-length string to a fixed-length string, or assign one fixed-length string's value to another. It can't assign a null value to a variable-length string except from another variable-length string. This missing capability is supplied by **StringRelease (B$STDL)**.

When assigning values to a fixed-length string, **StringAssign** evaluates the **LSET** function. If the source string is longer than the destination string, it is truncated. If shorter, it is right-padded with **CHR$(32)** characters to fill. I have yet to find an occasion when it was desirable to assign a value to a fixed-length string by using **StringAssign.**

B$ASSN is a horribly overloaded library procedure, most likely the product of "top-down" design. There is little code in common between the task of assigning a value to a variable-length string in string space and that of assigning a value to a fixed-length string in data space. Individually named library procedures for each of the several separable functions required would have been easier for assembly language programmers to use (not requiring otherwise unnecessary input arguments), would have produced better compiled program performance (the necessary parsing would have been done at compile time), and would not have suffered from **B$ASSN**'s logical lacuna.

StringAssign requires six one-word input arguments to be pushed on the stack prior to the CALL—three for the source string followed by three for the destination string. You must push the segment address, offset address, and string length (in that order) for each of the two strings. The address pushed for a fixed-length string is the far address of the first byte of the string. The address for a variable-length string is the far address of its descriptor, even though all descriptors are "near" double words in DGROUP; variable-length strings are identified by a zero for the string-length argument. (If **StringAssign** had been designed as a one-way function that assigned a value from a fixed-length to a variable-length string, only four one-word arguments would have been required.)

StringRelease assigns the null value to the designated string. It takes as its single argument the offset of the string descriptor in DGROUP. (Even though our one-way **StringAssign** function could have handled zero-length source strings, **StringRelease** would still have been useful.)

THE VLABEL FUNCTION

VLABEL.ASM (Listing 1), a function enhancement for BASIC 7.0, ensures that the user has inserted the proper diskette into the designated drive after a displayed request. VLABEL returns a string containing the volume label, if any, of the disk in the drive designated by the first character of the argument string. (A null argument string will return the label of the default drive.) A null string will be returned if the disk has no label or if the designated drive does not exist. A critical "Disk not ready" error will be returned to the calling BASIC program if the argument refers to a drive that doesn't have a disk installed or one that has an open door.

The routine's data segment consists of an Extended File Control Block (EFCB) used to find the label on the disk and the descriptor used by the function to return the string. The "old" *EFCB version of the DOS* **find first** function (AH = 11h) is used instead of the "new" ASCIIZ version (AH = 4Eh) because the new version usually inserts a decimal point into the label string and may remove intervening spaces.

A call to **StringLength** checks for a non-null input argument. A null argument (zero length) indicates that the current drive is requested; the null already in AL is used to load the drive code into the EFCB. Otherwise, **StringAddress** is used to find the argument in string space; the first character is moved to AL and converted to the DOS EFCB drive code (1 = A, 2 = B, etc.). Because the function may be called more than once in a single program, the 11 "?" symbols are reloaded into the EFCB before the DOS **find first** call. If no label is found, **StringRelease** is called to force a null descriptor for the VLABEL return.

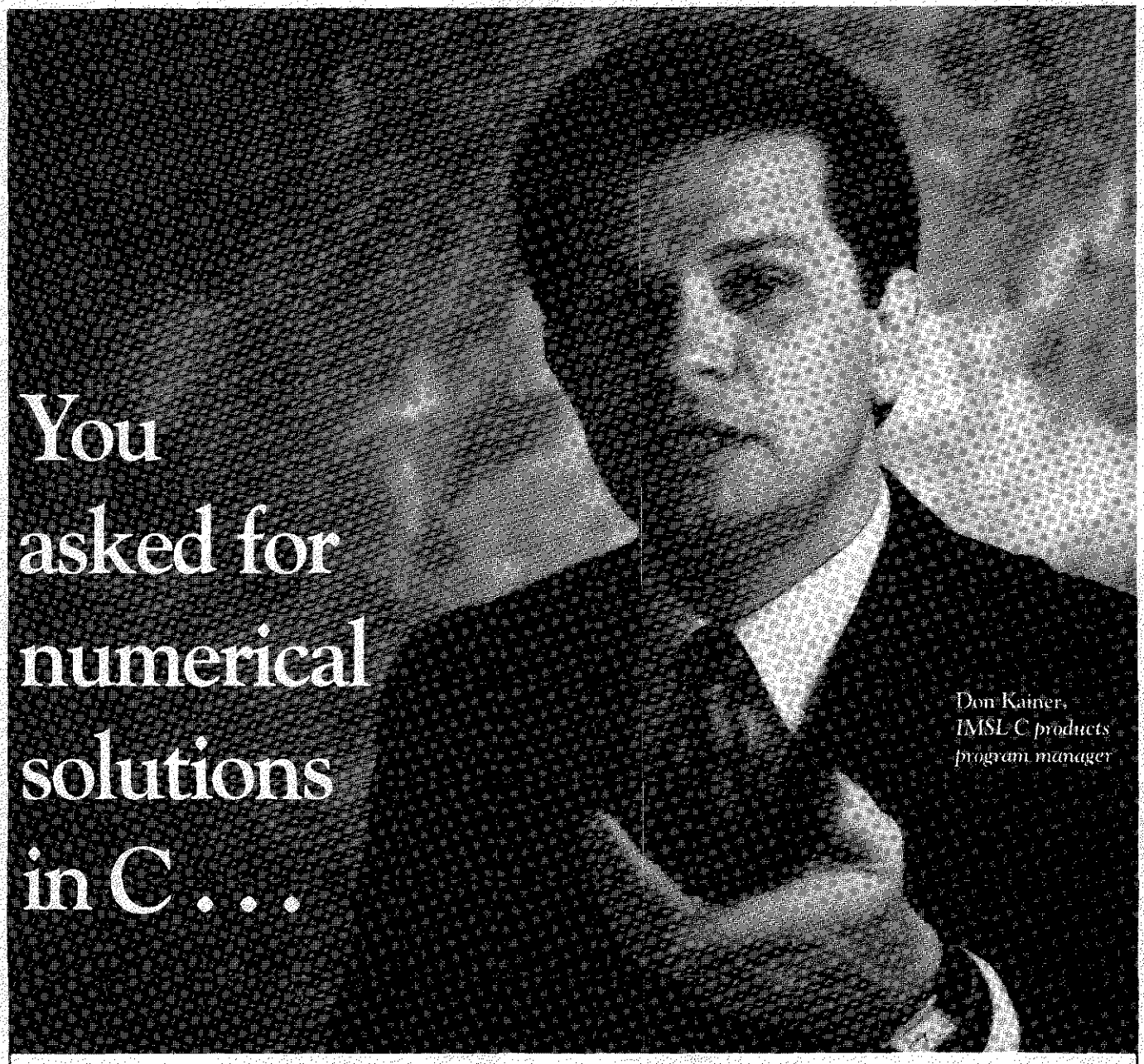
The little fiddle commented as *Decrease CX ...* is used to effectively delete trailing <SP> symbols from the found label. This code segment avoids two of the potential traps waiting for programmers using assembled string routines to be linked to BASIC programs.

There are restrictions on using the direction flag. The BASIC 7 documentation (*Programmer's Guide*, Chapter 12) cautions the programmer to "preserve the flags register" on the stack if the assembled routine "modifies the direction flag by using the STD or CLD instructions." This is both overkill and poor advice; all library string procedures in BC, at least since version 4.0, expect the direction flag to be clear (UP) on entry and leave it clear on exit. There is no need to push the flags register. You need to use the CLD instruction only if you used the STD instruction in the routine; you must clear the direction flag before any BASIC library procedure is called; this may be well before the return to BASIC. In VLABEL, the direction flag is cleared before calling **StringAssign**.

The other potential problem relates to routines that calculate the usable length of a fixed-length string; a zero must not be passed to **StringAssign** as the length of that string. The *Decrease CX* fiddle can never return a length of less than one byte, even if VLABEL finds a pathological disk label containing only <SP> characters. **StringAssign** will always treat a zero-length string as if it were a BASIC string variable, using the double word starting at the string address for the descriptor. A test driver for the VLABEL function is included on the code disk.

Advanced QuickBASIC users may want to upgrade to BC version 7.1. In addition to the added capability, the new BASIC compiler can produce much shorter (if not necessarily faster) executable code for most applications. Unfortunately, you'll have to pay street price because Microsoft doesn't offer low-cost upgrades of BASIC 7 to registered QuickBASIC users. PJ

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



. . . Introducing the IMSL C/Base/Library™ for C programmers

For more than 20 years IMSL® has been the leading provider of mathematical and statistical solutions with over 12,000 customers in 65 countries.

IMSL is proud to introduce its newest scientific software . . . The IMSL C/Base/Library™.

The C/Base/Library™ is a collection of IMSL's most frequently requested numerical functions covering a broad base of problems including *linear systems*, *optimization*, *differential equations*, *eigensystem analysis*, *statistics* . . . and much, much more.

Using IMSL's professionally designed C/Base/Library™ allows you to realize significant savings in the time and cost of your C applications programming and relieves you of the task of writing and debugging your own numerical code.

The IMSL C/Base/Library was designed by C programmers for C programmers. The140 exhaustively tested user-callable functions, along with optional arguments, give you greater accuracy, reliability and robustness.

And of course, behind every IMSL license is a team of product support professionals providing extensive documentation, product maintenance and consultation services to ensure value and performance in your IMSL software.

If you're looking to increase productivity and efficiency in your C programming, contact us today.

Telephone: (713) 782-6060;

Fax: (713) 782-6069;

Or write: IMSL, Inc. 2500 Permian Tower, 2500 CityWest Boulevard, Houston, Texas 77042-3020.

IMSL™
The numerical solution source.

Copyright © 1991 by IMSL, Inc. All rights reserved. IMSL is a registered trademark of IMSL, Inc. C/Base/Library is a trademark of IMSL, Inc.

Contact this advertiser directly.

PRODUCTIONLIBRARY.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Porting to the Mac

What you need to know to write a super application for the Macintosh

By W. Sumerlin

So you want to port your application to the Macintosh. As luck would have it, the Macintosh is one of the most difficult systems to learn and master. It takes six volumes to describe all of the Toolbox (operating system) calls. Although you don't have to know everything in these books, you do need a basic understanding of all the major system components and an intimate knowledge of some of the subsystems.

The rewards are a state-of-the-art application if you do the task right. If the idea you start with is good enough, you might even develop a super application, one that initially owns the market like Lotus 1-2-3 did on the PC.

TARGET AUDIENCE

Who is your target audience? If the application is for your personal use, you don't need a great product with a wonderful user interface and full documentation. But if your company needs an application to perform a specific function, or if you're trying to break into the commercial market, you'll have to do much more; the finished program must have a Macintosh user interface and be consistent with other major Macintosh applications. You need to provide final user documentation. And you will have to test your application thoroughly to screen out any major bugs.

For a commercial product, you need to add flashy packaging and full documentation. The presentation/orientation of your product must make potential customers want to buy and use your application. Finally, your product must be able to anticipate the most likely evolution of the marketplace.

Keep in mind that the Macintosh is a sophisticated and complicated environment. Learning the environment and becoming a good programmer takes years, not the weeks or months required by DOS. You also need access to other knowledgeable Macintosh programmers; their shortcuts and experience are a must. Without them, your chances for success are minimal.

A RANGE OF HARDWARE

What is your target environment? A wide variety of Macintosh computer systems, as well as peripheral and specialty hardware devices, are available. Generic software applications such as word processors and spreadsheet programs must be able to run on any Macintosh computer system, regardless of the model and configuration. As with most operating systems, the specific hardware differences are hidden from the user (but not from the programmer) by the Macintosh operating system.

Table 1 lists the Macintosh computer systems that Apple has manufactured and shipped. Note that not all of the systems listed are still available, though they do exist in the Macintosh installed base. The table also identifies the Motorola microprocessor and other pertinent system information, in case you have a performance-sensitive application that needs to take advantage of some specific hardware.

If you are going to develop a *generic* software application, be ready to support the application across all platforms—old *and* new. You can make recommendations about minimum and preferred hardware configurations, but be prepared for criticism if you don't support all the platforms.

THE BASIC MINDSET

With some exceptions, the Macintosh is exclusively an interactive environment. To initiate a program a user *clicks* with a mouse on a data file or an application's executable program file. Unlike other computer environments, the Macintosh requires a pointing device. Once a program is initiated, a user interacts with the Macintosh through a series of menus (Figure 1), windows, and dialog boxes. With enough memory

Bill Sumerlin is commonly found poring over piles of C code at MMC AD Systems, where they specialize in tools for C programmers on the PC, Macintosh, and Sun workstations. His address is MMC AD Systems, Box 360845, Milpitas, CA 95036.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

(2–4MB) you can also run MultiFinder, an Apple system program that enables the execution of simultaneous multiple applications.

In Figure 1, the menu bar at the top of the display identifies the basic functions available to the current, active application. The first three menu items are found in almost all applications. Many applications have keyboard equivalents that emulate selecting and clicking on menu entries. For example, simultaneously depressing the *command* and *O* keys would cause the disk to be opened and its contents to be displayed on the desktop. A great source for Macintosh tricks and shortcuts is *The Macintosh Bible*, edited by Arthur Naiman.

TARGET THE ENVIRONMENT

You can target your developed software product for any of several environments. The most common environment is an application program. An application program runs independently of all other programs. It performs the requested operations and either presents the results through the graphical user interface and/or stores them in a Macintosh file. If another program is going to use the stored information, it should be in the standard Macintosh file format for that category of software.

A **desk accessory** is a miniature supporting application. This Macintosh version of a TSR pops up on top of the application window that is currently active. Available desk accessories are automatically listed by the OS in the Apple menu and can currently be installed and removed only via Font D/A Mover, an Apple system utility program.

A **device driver** works with the operating system's Device Manager and a hardware device. It is typically incorporated inside an **init** program (see below), which is responsible for installing it when the system starts up.

A **code resource** is a software function that is stored in the resource fork of a file. The Macintosh operating system calls it in order to perform a predetermined task. One example of a code resource is a window definition function (WDEF); this function draws and manages a nonstandard Macintosh window for an application.

An **init** program is executed by the system during the boot process. To be executed, it must be placed in the current operating system's folder/directory. The **init** program also has to have the appropriate resources defined in the resource fork of its program. The Apple-supplied programs that manage the mounting and unmounting of CD ROMs are examples of **inits**. A number of **inits** do not work with one another, with some major applications, or with the current operating system revision level. Use them with care and be prepared to remove one or more if you have a problem.

A **cdev** program runs under the Control Panel, a system desk accessory that enables a user to control the computer system environment. For example, Figure 2 shows the **General cdev** program window that enables a user to set the speaker volume, the rate at which the cursor blinks, and the size of the disk cache, among other things. **Cdevs** are installed simply by placing them in the system folder.

FIGURE 1: Menu bar

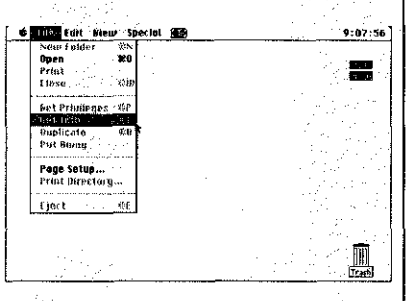
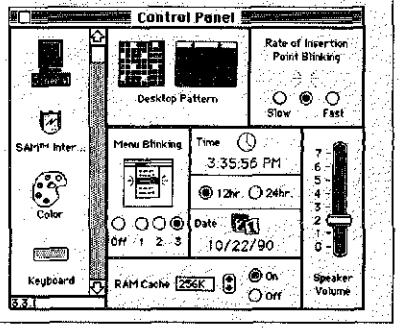


FIGURE 2: General cdev window



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Macintosh Programmer's Workshop (MPW) is the program development environment that is developed and supported by Apple. All of Apple's language compilers (e.g., C, C++, and Pascal) and supporting tools (e.g., linkers and librarians) run in this special environment. MPW tools can be executed in a batch fashion through a command line interface that is similar to PC DOS's, but more powerful.

In addition, multiple-level dialogs that provide fully interactive graphic user interfaces for any tool can be created. If your programming aid is designed to work exclusively with an MPW compiler or tool, then you will have to develop it in and for the MPW environment.

MACINTOSH COMPLEXITY

With Apple's release of the new System 7.0 operating system, the number of environments will decrease. For example, to use an application as a desk accessory, deposit the application in the system folder. This will eliminate current concerns about program or resource constraints.

If you are starting to feel a little lost and uncomfortable, you are not alone. Although the Macintosh provides a state-of-the-art user interface system that is easy to use, it is *not* easy to program. Many definitions use other terms that require yet more definitions. Unfortunately, no single source or location contains a definitive list of definitions. The most complete reference work is *Inside Macintosh* Volumes I–VI, from Addison-Wesley.

The *Inside Macintosh* series provides detailed narrative and reference material about the Macintosh operating system. The volumes are extremely thin on examples, however.

TABLE 1: Apple Macintosh basic computer systems

Model	Processor	Clock	Cache	PMMU	Min	Max	FP	Slots	Screen
Mac Plus	68000	7.83			1MB	4MB		0	B&W:512x342
Mac SE	68000	7.83			1MB	4MB		*	B&W:512x342
Mac Classic	68000	7.83			2MB	4MB		0	B&W:512x342
Mac Portable	68000	16			1MB	2MB		*	B&W:640x400
Mac SE/30	68030	15.667	256		1MB	8MB	68882	*	B&W:512x342
Mac II	68020	15.7		option	1MB	8MB	68881	6	
Mac LC	68020	16			2MB	10MB		*	controller
Mac IIx	68030	15.667	256	yes	1MB	8MB	68882	6	
Mac IIcx	68030	15.667	256	yes	1MB	8MB	68882	3	
Mac IIsi	68030	20	256	yes	1MB	17MB	option	*	controller
Mac IIci	68030	25	256	yes	1MB	8MB	68882	3	controller
Mac IIfx	68030	40	256	yes	4MB	8MB	68882	6	

Clock: Motorola processor clock speed in MHz.

Cache: Instruction and data cache in bytes (for improved performance).

PMMU: Paged Memory Management Unit. Required for A/UX (Apple's version of Unix) and System 7.0's virtual memory features.

Min/Max: Minimum and maximum memory sizes in megabytes. MultiFinder and System 7.0 require a minimum of 2MB. Many applications require from 4MB to 8MB. Macintosh Programmer's Workshop (MPW) and THINK C require a minimum of 2MB.

FP: Floating Point coprocessor (aka FPPU). Though a coprocessor is not required (all necessary floating point functions are in ROM), it can improve performance substantially. Numerical calculations are performed through Apple's SANE (Standard Apple Numeric Environment) library. Numeric precision will vary depending on your calculation methods.

Slots: Number of slots available for peripheral cards. There are two basic kinds: NUBUS (general-purpose peripheral) and processor direct (indicated by * above).

Screen: Older Macintosh computers usually come with a built-in screen. The Mac LC, IIci, and IIsi come with a controller that supports both color and black-and-white.

The most popular C development environments available are Apple's Macintosh Programmer's Workshop and MPW C, each priced at $150, and Symantec's THINK C, which sells for $249. Table 2 provides an overview of the positive and negative points of the two environments.

MPW is a stand-alone application program that creates a Unix-like program development environment. The program features a powerful command shell, aliases and shell variables, I/O redirection, pipes, shell scripts, command histories, and regular expressions. Through MPW's Commando facility, most command tools provide a graphical user interface for complete interactive operation and execution. If you are comfortable programming in Unix, you will find the transition to MPW reasonably easy. MPW C, Apple's C compiler, is an MPW tool that compiles C source code and generates object code that other MPW tools operate on. As C compilers go, it's about average.

Symantec's THINK C 4.0 provides the fastest C compiler available. It compiles *and* links at 40,000–100,000+ lines per minute, depending on the Macintosh platform. THINK C also incorporates a powerful multiple-window editor with a multiwindow source-level debugging system and some performance-tuning facilities. Unlike a batch compiler that simply lists errors and locations, THINK C stops compilation and puts the user at the source-code location of the detected error. With THINK C there is no link step. As each file is compiled, it is automatically relinked into the existing program. You can inquire about possible link problems such as undefined symbol references whenever you want.

MPW VS. THINK

Both environments provide everything you need to produce a working application. In many respects both are superior to anything available in PC-DOS. Some key questions might help you make your selection.

How big is your development team? If you have programming teams of more than three people working on the same software component, use MPW C. It is very difficult to manage people working on the same code in a THINK C

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

project or program. If you are working by yourself or with one other person, use THINK C.

Do you need batch or interactive operation? If you need batch operation, MPW C provides it, whereas THINK C does not. Both MPW C and THINK C provide interactive operation, although THINK C's is better integrated and more user-friendly. If you have large code files to load into the Macintosh environment, you can still use THINK C to compile them with minimal user interaction. But this minimal interaction assumes that your code contains only a few bugs for manual correction.

Do you need some sort of code control, accounting, and/or audit trail? Only MPW C offers these functions.

COMPILER CAUTIONS

Both manufacturers state that their C compilers are not ANSI-compliant, and both list the known areas of noncompliance. But because neither of the compiler manufacturers feels compelled to be ANSI-compliant, they have not tested their compilers enough to be aware of all of the noncompliant areas.

The size and content of the runtime libraries are extremely limited for both compilers. Their libraries are missing functions that might be considered standard. You also need to be careful that a library function operates in the expected fashion. Even though both manufacturers have made substantial improvements in their latest compiler releases, we have had so many problems in the past that we use very few runtime library functions today.

As with all computer system environments, use care when working with floating point numbers, especially if numeric precision is a consideration. Table 3 summarizes the size and value ranges for the various data types.

THINK C and MPW C have inconsistent **int** and **double** types. THINK C uses 16 bits for **int**s and MPW C uses 32 bits. THINK C also adds a new nonstandard keyword, **extended**, to represent the **long double** data type. There is also a difference in precision for the largest data type, depending on the presence of an FPPU. The FPPU uses 96-bit extended format since the 68020

TABLE 2: MPW and THINK

MPW C Positives:
- Shell and batch-processing environment
- Easily extensible environment

MPW C Negatives:
- Mediocre C compiler
- Weak source-level debugger

THINK C Positives:
- Super-fast compiler
- Well-integrated environment
- Good source-level debugger

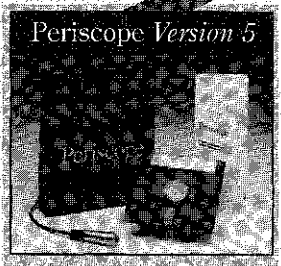
THINK C Negatives:
- Environment is not user- or third-party-extensible
- Can't get a composite list of syntax errors
- Has no facility to define external symbols to control compilation. At least one common include file must be modified.

Bugs are *expensive.* Can you afford them?

Richard Fink, President of RainTree Computer Systems, writes, "...What it [Periscope] offers is probably the most comprehensive debugging capability on the market today. And for you and me, that means getting to market sooner. Getting to market with a cleaner product. That's an objective we all know about."

Periscope handles the level of debugging you need. Whether you're developing applications written in a high-level language, doing low-level system development, or something in between, Periscope can help you find the bugs. Randy Brukardt, a developer of the Janus Ada compiler, writes, "I couldn't imagine using anything else...It is just as useful debugging my Ada code at the source level as it is for finding bugs in assembler code, even TSRs and device drivers."

There's just not much you can't debug with Periscope. For example, you can debug device drivers and TSRs, child processes, and software interrupts. You can trace DOS and debug foreground and background programs in the same session. Large programs are no problem. Periscope supports Plink and .RTLink overlays, and Windows 3.0 programs in real mode. You can monitor software running on another system. And you can debug the boot process, hardware interrupts, and real-time code. The Periscope software runs on 8088 through 80486 machines, supports 80386/80486 debug registers, and runs with 386 control programs in the system.



Periscope Model II includes a break-out switch and the new Version 5 software. The new software, included with all models, features a menu system that makes Periscope easier than ever to learn and use.

There's a Periscope model for every budget. Prices start at **$195** for software-only Model II-X. Model II with its handy break-out switch is **$225.** Model I with 512K of write-protected RAM is **$595** for PCs and **$695** for PS/2s. Model IV with its real-time hardware trace buffer and breakpoints is **$1895** to **$2395**, depending on your processor and its speed. We'll be happy to help you decide which model you need.

Start saving money today. Call Toll-Free: 800-722-7006

Overseas, call: UK - Roundhill Computer Systems, 0672 84 638; Germany - H/B EDV, 07542 6038; ComFood, 02304 2093; Sweden - LinSoft, 013 124 790; Denmark - Rosenholm Computing, 02 88 72 39; Australia - BJE Enterprises, 02 858 5611.

The Periscope Company, Inc.

1197 Peachtree St., Atlanta, GA 30361, USA • 404/875-8080 FAX 404/872-1973

Circle 39 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

accesses data more efficiently on long-word boundaries. For both compilers, you must set a compile option to use the FPPU. Though the Macintosh O/S provides a way to determine the presence of an FPPU, there are no means to support both 80 and 96 bits of (extended) precision; this is a compile option.

When performing expression evaluation, THINK C temporarily converts all floating point numbers to **doubles**, whereas MPW C converts only to the highest precision of the current two operands being evaluated. Thus, you can get different numeric results when a program has been compiled by the two Macintosh C compilers.

AN ARRAY OF PROBLEMS

Array sizes can also be a problem. Both compilers support the Macintosh as a segmented memory architecture using 16-bit offsets. As a result, you can't port an application from a virtual memory architecture that uses large, static arrays and manage to compile it in either environment. That's not surprising—array sizes have been a problem ever since they invented segmented memory architectures.

The above problems are all manageable: if you port code between environments, be sure to work with the minimum common denominator. For example, if you work in a Unix environment, you can ignore the ANSI standard. Thus, use great care when selecting which C syntax to use (e.g., no function prototypes, so you need a **lint**). Because the Macintosh Toolbox (O/S) is rich in functions and is directly available to the C programmer, neither manufacturer expends more than minimal effort on the runtime libraries.

Once you have selected a Macintosh development environment, you should view the other one as a foreign computer environment. In addition to relatively minor C dialect differences, there are major differences in system **include** files (content and name), in floating point calculations, and in Macintosh O/S calling sequences (function names and argument types).

One of the more interesting bugs in Apple's MPW C 3.0 compiler involved incorrect code generation when the ++ and -- operators were included in expressions. Although Apple is now selling/shipping version 3.1, which fixes most of the bugs, many customers, including ourselves, still ship products that were built with version 3.0. Why? Because Apple's solution to the ++ and -- bug had a significant impact on program performance. Their quick fix adversely affected their compiler's code optimization. So we changed the source code instead of the compilers to maintain our current program performance levels.

THINK C 4.0 contains support for object-oriented programming (OOP). Their version of C is a close subset of C++. In addition to language support, there is a complete object library with source that supports the general Macintosh user interface. If you are interested in learning and/or trying OOPs, THINK C and its object libraries and examples provide an excellent vehicle for the attempt.

TABLE 3: Values and data ranges

MPW C Type	Bits	Minimum	Maximum
int	32	2.14E-9	2.14E+9
float	32	1.5E-45	3.4E+38
double	64	5.0E-324	1.7E+308
extended **	80	1.9E-4951	1.1E+4932
THINK C Type	**Bits**	**Minimum**	**Maximum**
int	16	-32,768	32,767
float	32	1.5E-45	3.4E+38
short double	64	5.0E-324	1.7E+308
double **	80	1.9E-4951	1.1E+4932

**NOTE: becomes 96 bits if FPUU used

OTHER DEVELOPMENT TOOLS

If you are still serious about programming on the Macintosh, you should join APDA, Apple's Programmer's Development Association. Through it you can buy Apple's software products such as MPW and MPW C, Apple Macintosh technical notes/information, and third-party software products.

If you have gotten used to a plethora of tools in your current development environment, get ready to be disappointed: not many programming tools and aids are available in the Macintosh environment. For C support tools such as cross-referencing, lint, and pretty-printing, our own C Programmer's Toolbox/MPW ($295), McCLint ($149.95), and McCPrint ($99.95) programs are without competition—that is one of the advantages in developing for the Macintosh market. Some sources for identifying programs can be found under "Resources" (p. 65).

GRAPHICAL USER INTERFACE

To be accepted by existing Macintosh users, your program must have a Macintosh graphical user interface. If your application runs stand-alone, you have several choices for creating/developing an interface.

Using a **homegrown** interface will require that you create code for all your own menus, windows, dialogs, etc.—a very difficult and time-consuming task. I highly recommend obtaining some third-party code that already works and modifying it to meet your needs.

Third-party **linkable libraries** are available. These can provide most of the code that you need. For example, Professional Programmer's Extender ($395) or MacInterface ($295) provides menus, windows, and printing routines in addition to support for MultiFinder.

Prototyper ($299) and AppMaker ($295) are **GUI** generators that allow you to define and create your program's user interfaces interactively. These CASE programs can then generate the C source code necessary for you to create and run your defined user interface.

For the **object-oriented** approach, THINK C 4.0 includes object support as well as a complete class library that supports the basic Macintosh graphical user interface. Apple's C++ with MacApp also provides a set of object libraries to help build and manage your user interface. Either of these vehicles probably will require some modifications to the structure of your program. Note as well that your finished programming product will not be portable to any other environment.

Before selecting a method to implement the graphical user interface, identify generally what your interface will look like, how it will operate, and what interface primitives will be required. The Macintosh has a number of simple and complex primitives, including the following:

- **Screen layout:** windows, dialogs, and alerts.
- **Controls:** buttons (check box, radio), text (static, edit), icons (system, user), pictures, lines, rectangles, palettes, lists, scroll bars, resource controls, user items, and more.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

- **Menus**: menus (pop-up, tear-off, hierarchical, etc.), menu bars.

Not all interface primitives are supported by the implementation methods.

Apple defines a set of rules that all applications are supposed to follow. The rules can be found in *Inside Macintosh* and in a number of technical notes. When you think you're finished with your interface, be sure to take the time to go back and look at the rules again. If you don't get the graphics user interface right on the Macintosh, there is absolutely no way that your application will be successful.

DEVELOPMENT HINTS

You will find that there are several different ways to approach and solve a development or programming problem on the Macintosh. This flexibility may let you find a solution that works for your particular problem, but though the solution works with your current system, compiler, and development environment, there is no guarantee of its working with future releases. Hence, it is important that you find and use mainstream solutions. As the system, compiler, and environment evolve, the various manufacturers will collaborate to keep the mainstream solutions functioning, but other solutions may cease to work.

Because we didn't know any better when we started, we suffered mightily along the way. Some Macintosh folks think of it as paying your dues. The following suggestions come from our own experiences in developing, publishing, selling, and supporting three Macintosh products.

For personal applications, select an easy project. It's preferable to start with source code that already works in your chosen development environment. Find it in books (e.g., *Using the Macintosh Toolbox with C* [Sybex, 1986]), magazines (*MacTutor*), and on bulletin boards. Remember that source code that can be compiled and executed in either MPW C or THINK C may not compile successfully or work in the other environment. Your objective is to modify the source code in some way, primarily learning the development environment and how to work with the Macintosh Toolbox.

For initial projects, think small and stay focused. The Macintosh is an easy environment to get diverted in, and any diversion can get you lost.

If you're developing group applications or you're under time constraints, find someone knowledgeable in Macintosh applications. But be forewarned: good people don't come cheap. Skilled Mac programmers are scarce, although a number of people may claim to be qualified. Always check references and look at completed projects.

Given budget constraints and concerns, think twice about porting your code to the Macintosh. If you don't do the job properly, your application will not sell and will not be used. A current Macintosh user can be a tough sell, but once sold, they are loyal customers.

SOME FINAL NOTES

So what do I use? I program principally on a color Macintosh II with an FPPU and a PMMU. Although I can choose to work on the latest PCs and Suns, I prefer the Macintosh. I don't always use the latest and greatest systems and/or applications. I have a tendency to stick with what works and what I'm comfortable with.

I use the THINK C 4.0 compiler for most of my primary software development (not the OOP extensions) and then port to the other environments (DOS and Sun). I also use MPW and MPW C, but only in a supporting role. For communications in and out of the Macintosh environment I use Apple's network between Macintoshes and PCs, TCP/IP (FTP and Telnet) to other computer systems, and principally Kermit over phone lines.

Note well that I do *not* wait until the end of a project to find out whether something will port. I try things out in all target environments throughout the process.

And finally, to make explicit a point that is implicit in all this: it is imperative that you know your competition—especially what the product *looks* like—before you write a single line of code. If you don't create a significantly better product, no amount of marketing or sales activities will make your application sell better.

VENDOR INFORMATION

AppMaker
Bowers Development
P.O. Box 9
Lincoln Center, MA 01773
(508) 369-8175
Circle 217 on Reader Service Card

MacInterface
Holder, Egan & Company
4148 Spring Hill Road
Midland, MI 48640
(800) 782-9976
Circle 218 on Reader Service Card

Professional Programmer's Extender
Masters Publishing
3571 Pheasant Run, #7
Ann Arbor, MI 48108-2460
(313) 971-1118
Circle 219 on Reader Service Card

McCLint; McCPrint; C Programmer's Toolbox
MMC AD Systems
P.O. Box 360845
Milpitas, CA 95036
(415) 770-0858
Circle 220 on Reader Service Card

Prototyper
SmetherBarnes
520 SW Harrison, Suite 435
Portland, OR 97201
(503) 274-7179
Circle 221 on Reader Service Card

THINK C
Symantec Corporation
10201 Torre Avenue
Cupertino, CA 95014
(800) 441-7234
Circle 222 on Reader Service Card

RESOURCES

- *Macintosh Development Tools & Languages Guidebook, 1991 Edition*, and
- *Apple Engineering/Scientific Solutions Guide, Winter 1989/90*. Both are annual catalogs from Apple Computer, Inc.
- *Inside Macintosh*. Volumes I–VI (Addison-Wesley Publishing Company).
- *The Macintosh Bible*. Arthur Naiman, Editor (Goldstein & Blair).
- *MacTutor*. Macintosh's monthly programming magazine; (714) 777-1255.
- *Macintosh Buyer's Guide* or *Macintosh Product Registry*. Quarterly catalog published by Redgate Communications Corporation.
- Apple's Programmer's Development Association; (800) 282-2732.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Disassembler roundup

Theory, practice, and products for those who get into code

By Brett Glass

Some time ago, computer columnist Jerry Pournelle received an IBM PC peripheral board in the mail. There were no markings on the board to indicate its purpose, no documentation in the package, and no return address. Mystified by the new arrival, Barbara (now Heather) Clifford, Jerry's editor at *InfoWorld*, asked if I could identify the board.

I noted that the board contained only two decoder chips and an EPROM and presumed it was simply a ROM board that added some function to an IBM PC. Fortunately, I decided to remove the EPROM and pass its contents through a disassembler before plugging it into my own system.

Disassemblers—programs that attempt to make sense out of machine code and provide some form of workable source—are tools of desperation. There's seldom any reason to use one if you have a program's source or if you understand how it works. But what if the source code to a program has been lost or destroyed by a disgruntled former employee? Or if some of your program files are infected by an as-yet-undiscovered computer virus? Or if you suspect that a file uploaded to your BBS is really a Trojan Horse? Or if a bug in your BIOS or DOS appears to be crashing your system? In all of these cases, the only solution is to cut the Gordian knot with a disassembler.

Brett Glass works as a free-lance writer, programmer, and consultant who lives in Palo Alto, California. He can be contacted at rogue@well.sf.ca.us (Internet), at 7226,3673 (CompuServe), or as glass on Bix.

A TROJAN BOARD

I used an EPROM programmer to read the code in the mystery board, and then I passed it through the primitive disassembler in DEBUG. I first discovered that the ROM contained a special signature that would cause the BIOS to run the code inside when the machine booted. This was expected; most peripheral board ROMs are executed, at least in part, at start-up. But the code that followed was startling: it was designed to seek out the partition table of each hard disk attached to the machine and scramble it, leaving the disk unusable.

Most Trojan Horse programs are carried in software and downloaded from bulletin boards by careless callers. Here we had something totally new: a booby-trapped program that had been embodied into the hardware. In other words, Jerry may well have received the very first Trojan Board!

Fighting rogue programs isn't the only use for disassemblers. Whenever you have an executable program but need to patch, enhance, or understand it without the source, a disassembler is the tool for the job. In this article I'll cover the basics of disassemblers and survey several popular products that run on the PC. All are more powerful than the disassembler in DEBUG, and some are quite good at second-guessing crafty compilers and clever assembly-language coders. Considering the difficulty of the task at hand, this ability is quite impressive.

WHY THEY CALL IT CODE

Disassembling any nontrivial program that has been compiled to the tight, quirky, and ambiguous machine language of the Intel 8088 is difficult. Many instructions can be written in two different ways, and a single physical mode address can be represented by thousands of combinations of 16-bit segments and 16-bit offsets. Most programs use one of the "standard" memory models (tiny, small, compact, medium, large, or huge), but even with these there are hundreds of possible combinations and variations.

Memory can be a crazy quilt of data and code interleaved at random. Worse yet, the distinction can be obliterated by self-modifying code or code that uses its own opcodes as data. In order to reconstruct a program, a disassembler needs to unravel the mystery and separate the code from the data, inasmuch as that is possible.

Tables that contain destinations for jumps and calls add another twist to the

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

disassembly puzzle. Many compilers implement case or switch statements with jump tables, but even the most intelligent disassemblers have trouble distinguishing the tables from other data. For instance, in one of the programs used in our test, ANSI.SYS, the CPU loads a 16-bit immediate quantity into the BX register. It then does some calculations and memory references and eventually executes the instruction **jmp ax**. However, not a single one of the disassemblers that I tested was able to figure out what relation the original constant bore to the jump address. (The answer: the constant was the address of a location three bytes below the beginning of a table of records. Each entry in the table contained three bytes: a character to be matched and a jump address to go to if a match occurred.)

Vectors that change on the fly are even harder to follow. A few of the disassemblers I tested could find interrupt vectors by watching the register values during calls to DOS's Get/Set Interrupt Vector functions, but all of the tested disassemblers were confused by the simple vectored jumps in the I/O section of the Turbo Pascal 5.5 runtime library.

FILE COMPRESSION

As if all this isn't confusing enough, file compression utilities add yet another layer of obfuscation to many programs. The Microsoft Linker can pack EXE files to save space and load time, as can the separate EXEPACK utility program. Some of the utilities within MS-DOS and PC-DOS (such as PRINT.COM) are processed with another utility that packs an EXE file into a COM file (again with a built-in unpacker that enlarges it at runtime). This is called a "hidden EXE file" or a "multisegment COM file." Some utilities, such as the Freeware program LZEXE, squeeze the code still tighter by performing Lempel-Ziv compression on the file. Needless to say, if you load one of these files the only code that's likely to be readable is the unpacker itself—so it's important to have a utility that can undo what the packer has done.

SOME RULES OF THUMB

Only a programmer can truly "psych out" another programmer. Perhaps this is why disassembling 808X programs is an art form, not a science. It's extremely difficult to codify all the subtleties of the art. Still, here are some useful rules of thumb you must take into account when deciphering a program. If your disassembler "understands" them, so much the better.

The starting point of the program is always known. A COM program is loaded at address 100h in its code segment; execution begins with the first byte of the file. The header of an EXE file gives its entry point, as well as "fixups" that point out the relocatable addresses. This information lets a disassembler pick up the thread of execution.

If there aren't any branches in an instruction sequence, it's usually safe to assume that one instruction follows another until you get to a **call**, **jmp**, conditional jump, or **ret**. Because execution can't continue past a **jmp** or **ret**, whatever follows might be code or data; there's no way of knowing without more investigation.

Some instructions and prefixes are uncommon. These include the **lock** prefix (which doesn't do much on the PC architecture), **hlt** (which halts the processor), and **esc** (which is part of the 8088's coprocessor interface). If these prefixes are present, they're probably data being misinterpreted as code. The same is true of the instruction **mov [bx+si],al**, which happens to be encoded as two bytes of zeroes.

Scanning for sequences of printable ASCII characters is a good way to find a program's data areas. In one of the programs I used, VTREE.COM, scanning revealed nearly all of the pre-initialized data in the program.

Disassemblers have a hard time finding jump tables, but humans are very good at it. Look for strings of 16-bit or 32-bit pointers that increase uniformly and/or that are close together. (They should all point within the program, at any rate.)

Watch the parameters that are passed to DOS calls. For instance, if DOS function 9 is called, DS:DX should point to a message to be emitted at that point in the program.

When compiled programs use multiple code segments, they tend to break across the module boundaries. Look for routines with related functions within a single segment. Also, be on the lookout for unusual uses of the stack. Most disassemblers track register values, but some don't watch for sequences in which a value is pushed and later popped into a different location. One sequence that really confuses most disassemblers is to **push** a new CS, **push** a new IP, then **ret far** to jump to the pushed address.

You don't always need to understand the entire program to get the job done. If you're looking for the video routines in a BIOS, for instance, disassembling the entire ROM is overkill. That may be a good thing; many BIOSes

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

are between 32 and 64K of tight, machine-dependent code.

Finally, comment your work. All but one of the products in this review allow you to add comments that remind you of the discoveries you've made. This can be done either by directly modifying a screen display or by changing a definition file.

THE TWOFOLD PATH

The disassemblers I looked at fell into two groups: batch mode and interactive. A batch-mode disassembler processes a program from beginning to end with no user interaction during disassembly. (If you can affect the output at all, it's by modifying a control file between passes.) Interactive disassemblers let the user experiment with and update the assembly listing on the fly.

From half a dozen disassemblers, I picked two from each category for a closer look. The batch-mode disassemblers were **Sourcer** and **DISnDATa**. The two from the interactive category were **Dis•Doc Professional** and **The Masterful Disassembler**.

To test the disassemblers, I used several programs of different types. The first (and simplest) was a program by Charles Petzold called VTREE.COM, which displays an image of a directory tree on your screen. The next two test cases—XMSDUMP.EXE (see Kim Kokkonen's article on p. 72 for the Pascal source) and WC.EXE—were both Turbo Pascal programs. XMSDUMP makes extensive use of inline procedures, and WC (a word counter) does simple text and file processing.

I tested device-driver disassembly with ANSI.SYS, the screen driver from PC-DOS 3.30. Also, if the program was designed to disassemble BIOSes (one in each category was), I tried doing so. I also ran all the programs on a small but deadly piece of code called TORTURE.EXE (see Listing 1), a program containing a few simple code sequences designed to befuddle debuggers and disassemblers. TORTURE was packed by the Microsoft linker, and WC by EXEPACK.

Finally, to test their ability to handle large files, I attempted to run each disassembler on the 150K-executable of Borland's MicroStar (from the Turbo Editor Toolbox).

THE UNPACKERS

I tried several unpackers for the packed EXE and COM files. V Communications sells UNPACKER, which contains two utilities—UNPACK and COM2EXE—to handle these files. Both executed flawlessly. The Masterful Disassembler came with a freeware program, UPACKEXE, which is part of Fabrice Bellard's LZEXE utility package. It handles packed EXE files but not multisegment COM files.

Dis•Doc Professional came with its own UNPACK utility, designed to work on packed EXE or multisegment COM files. DISnDATa claimed to work on packed files with no conversion, unpacking them automatically. It worked properly with WC, which was compressed by EXEPACK, but failed with TORTURE, which was compressed by the Microsoft Segmented-Executable Linker, Version 5.01.21.

SOURCER'S SCORES

Of the batch-mode disassemblers, Sourcer is the best known. It was the slowest product I tried, but also the most accurate—mainly because it partially simulates CPU activity to guess register contents before each instruction. Because it isn't a full simulation, it doesn't always get things right.

Sourcer starts with the assumption that everything is data, then revises that assumption as it finds references to

LISTING 1

```
; TORTURE.ASM -- Copyright 1991 by L. Brett Glass
; This "disassembler torture test" is designed to make
; life difficult for disassemblers and debuggers.

        .MODEL LARGE

CODE    SEGMENT PARA PUBLIC
        ASSUME CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING

; Execution starts here. The program executes a bunch of NOPs, followed by
; a string of ADD [BX+SI],AL instructions. The stack is then moved over the
; area, and execution is transferred to the REAL program via a push-ret
; sequence. The intent is to throw the disassembler off the scent.

INIT:
        PUSHF                   ; Push flags for upcoming IRET
        XOR BX,BX               ; Clear BX and SI
        XOR SI,SI
        PUSH CS                 ; Move CS to DS.
        POP DS
        MOV AX,OFFSET STACKSTART ; Get new SP
        REPT 256                ; 256 NOPs
        NOP
        ENDM
        REPT 255                ; 255 copies of ADD [BX+SI],AL
        ADD [BX+SI],AL
        ENDM
        XOR  AX,SI              ; XORing with zero does nothing, of course!
        ADD  AX,BX              ; Same with adding zero
        PUSH AX                 ; Push new SP
        PUSH CS                 ; Push new SS
        CLI                     ; Turning off interrupts can cause some
                                ; debuggers problems
        POP SS                  ; Set up new stack OVER this code!
        MOV AX,SEG REALCODE     ; DEBUG automatically executes a second
                                ; instruction after SS is loaded, which
                                ; will hide this load. The user will just
                                ; see SP being popped.
        POP SP
        PUSH AX                 ; Get destination segment onto stack
                                ; below flags.
        PUSH BX                 ; By now, a disassembler may have "forgotten"
                                ; that this is zero. We're jumping to offset
                                ; zero in the segment REALCODE.
        IRET                    ; Pop initial flags into flag reg, then jump
                                ; to start of "real" code. Interrupts are
                                ; turned back on here when the code is run.
                                ; But a debugger, which must turn interrupts
                                ; on between every pair of instructions, will
                                ; turn them on earlier and trash this opcode!
STACKSTART:

CODE ENDS

REALCODE SEGMENT PARA PUBLIC
    ASSUME CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING
    MOV AX,SEG DATA
    PUSH AX
    POP DS
    MOV DX,OFFSET Message  ; Prepare to print message
    MOV AH,9               ; Print message through DOS
    INT 21h
    MOV AX,4C00h           ; and EXIT
    INT 21h
REALCODE ENDS

DATA SEGMENT
Message  db 'Disassembler test',0Dh,0Ah,'$'
DATA ENDS
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

code. In many cases, instructions such as **nop** and **mov [bx+si],al** will be shown as **db 90h** and **db 0,0.** As it disassembles the program, Sourcer creates a file with the extension SDF documenting the types and labels it has assigned to code and data. If you are able to deduce things about the code that Sourcer can't, you can usually persuade Sourcer to see things your way by creating a DEF file that tells it how to view code and data.

To do this, you copy the SDF file to another file with the extension DEF, then add and modify control statements within that file to change the way the program is disassembled. Sourcer will heed the changes most of the time. (In a few cases it can be infuriatingly stubborn, refusing to take your advice over its own interpretation of the program.)

TRY, TRY AGAIN

After you modify the DEF file, you must run Sourcer again in order to see the new output. If Sourcer makes new discoveries as a result of the things you put in your DEF file, they'll appear in the listing and in a new SDF file. Unfortunately, some of the comments in your DEF file, as created by Sourcer, will become wrong on subsequent runs. (They'll contain labels that Sourcer assigns differently from run to run.) If you try to solve this problem by copying SDF to DEF again, you'll run into a different problem: many of the macro-like instructions you used to tell Sourcer how to interpret data will have been "expanded."

Sourcer generates long listings and assembly-language files that are normally 132 columns wide. To handle this output, you need virtual memory as well as a text editor with multiple windows (so that you can see the DEF and output files at the same time). A 132-column display mode is also preferred. If Sourcer sees something resembling a copyright notice, it puts it on every page of the disassembly listing.

In disassembling VTREE, Sourcer got all the code and data areas correct but did not show all the ASCII as ASCII (as DISnDATa and Dis•Doc did). It also didn't identify references to the PSP as such. In general, however, performance was very good. When disassembling XMSDUMP, Sourcer correctly identified Turbo's interrupt service routines for critical errors, divide by zero, etc. It did a fairly good job of disassembling the snaky code of the runtime library (RTL), except for a couple of jump tables it could not find.

Sourcer performed quite admirably on ANSI.SYS, labeling the device-driver header correctly. Unfortunately, the jump table problem cropped up once again, causing some code that was referenced via jump tables to be misinterpreted as data.

BIOS AND PACKED FILES

BIOS performance was outstanding. With its extra-cost BIOS Preprocessor, Sourcer is still the all-time champ at disassembling BIOSes. If you're interested in what's in your ROM chips, this is the product of choice.

When I attempted to run it on the packed EXE file WC.EXE, Sourcer correctly reported that the file was packed and that output quality would be "low" if I proceeded. I was curious about the unpacking code itself, however, so I told Sourcer to continue and got a glimpse of it. I then ran WC through the extra-cost UNPACK utility and disassembled it. The results were similar to those for XMSDUMP: no major problems except with jump tables.

On TORTURE.EXE, Sourcer required a minor correction to the DEF file to help it see the code as code. (Long runs of **nop**s had convinced Sourcer that it was looking at data.) However, it did correctly identify the segments.

Sourcer could not disassemble all of MS.EXE at once, though it's supposed to use EMS memory if available. Apparently my 2 MB was not enough. Had I really wanted to complete the disassembly, I could have broken the file into address ranges and disassembled each in turn.

Sourcer's 140-page perfect-bound manual describes the product's operation in detail but is difficult to keep open on a desktop. It would help to have more examples of how to use Sourcer's somewhat terse definition language to influence the disassembler's output. I floundered at first because I did not find many examples to copy.

All in all, Sourcer does the best analysis of any of the disassemblers I tested and gets extra points for handling BIOSes and packed files (albeit at an extra charge). The most infuriating things about this product are the long cycle time when you're working to disassemble code and the fact that the disassemblies are almost good enough to be worth it. If V Communications provided an interactive front end, Sourcer would have no peer in the industry.

DISNDATA'S PERFORMANCE

Like Sourcer, DISnDATa is a batch-mode disassembler: it runs to completion before you see the output. But unlike Sourcer, which offers DEF files to let you modify the output, DISnDATa expects almost no input from the user. There are no command line switches and only one parameter: the name of the file to be disassembled.

In my tests, DISnDATa asked for help in only two situations. When it found a "computed" jump (i.e., a jump through a jump table), it wanted to know the possible addresses it might jump to. (Alas, finding this information requires that you have a listing in hand. You need to hit RETURN, look at the output, and rerun the disassembler after you know more about the program.) DISnDATa also asks whether your program has multiple entry points. (Device drivers and some other programs do; otherwise, you simply hit RETURN when you see this question.) DISnDATa writes its output to the file you specify.

DISnDATa exhibited widely varying performance on my tests. It produced the best listing of all the tested disassemblers the first time I ran VTREE on it, but it didn't fare so well on some of the other tests. When it was good it was very, very good—but when it missed the mark there was no way to guide it via its limited prompts. When DISnDATa mistook code for data, I could not make it generate code for those sections of the program.

DISnDATa was able to disassemble VTREE.COM beautifully on the first try—better than Sourcer. It recognized certain ASCII strings, such as a short one containing the characters "*.*", when Sourcer did not. Its performance on ANSI.SYS was much worse than Sourcer's, however. Although I'd completely disassembled the file before and could give exact answers to every question about indirect jumps, DISnDATa

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

couldn't find some routines—and I had no way of telling it where they were.

AUTOMATIC UNPACKING

One nice feature of DISnDATa is that it automatically unpacks EXE files on the fly. Alas, it was the only program that complained that the linker-compressed file was faulty when given the program TORTURE.EXE. I used the freeware unpacker UPACKEXE to unpack it. Working on the unpacked file, DISnDATa could not follow the flow of execution when the program used an **iret** to do a far jump. (Sourcer had trouble with this as well, but I was able to set it straight.)

DISnDATa had no trouble unpacking WC.EXE, which was compressed with EXEPACK. The parts of WC and XMSDUMP that were translated correctly were clean and readable—again, better than Sourcer. However, DISnDATa translated runtime library routines as data rather than as code. Like Sourcer, DISnDATa ran out of room when attempting to disassemble MicroStar. DISnDATa does not have a BIOS disassembly function.

Although DISnDATa's first-pass performance was only slightly worse than Sourcer's overall (and in fact better on a couple of programs), its overwhelming drawback is its inability to accept additional input. Until computers have the intellectual capacity to out-strategize humans, disassemblers need to accept "hints" to correct the inevitable errors of interpretation that crop up during disassembly.

DIS•DOC'S RESULTS

Dis•Doc Professional is the first of the two interactive disassemblers I tested. When you use Dis•Doc you scan a disassembled listing in a screen window (see Figure 1), improving it on the fly until it's close to 100% correct. You also add comments, descriptive labels, and data types to document your work.

Dis•Doc scans the file when it's loaded and comes up with an initial listing that's not as accurate as Sourcer's but still surprisingly good. From that point on, it's up to you. Using a mouse or the keyboard, you can highlight areas of the listing and change them from code to different types of data or vice versa. Dis•Doc gives you a considerable amount of control over screen display, letting you include or exclude automatic comments of several kinds. It also has pull-down menus, a smorgasbord of formats, and a built-in patcher that allows you to patch code without reassembling it.

Dis•Doc's manual deserves applause for one key feature: it comes in a ring binder that lies flat on a desk. Alas, the text inside simply explains the program's menus and function keys, with only a brief tutorial. I'd opt for more examples and more practical advice on how to use the product.

Interactive disassemblers depend as much on the user as on their own algorithms for the production of quality disassemblies. Thus your opinions of these products are likely to be more subjective than they would be for batch-mode products. I was able to disassemble all of the test programs with little trouble, although I had to change data to code in quite a few places. Still, some programs—such as ANSI.SYS—came out at least as well in Dis•Doc as in Sourcer. And, of course, the process of "correcting" the disassembler was much easier than for Sourcer. (The same was true for The Masterful Disassembler, to be discussed below.)

Both Turbo Pascal programs—WC and XMSDUMP—disassembled quickly. The unpacker worked on the packed files with no problems. Dis•Doc also got started disassembling the BIOS in record time, though with fewer helpful comments than Sourcer. TORTURE required forcing a few areas to code (as for Sourcer), but the interactive process took less than 20 seconds. One thing that surprised me was that Dis•Doc, like The Masterful Disassembler, was able to handle large programs such as MicroStar without running out of memory. Because of this ability, they're much better choices for large, monolithic programs.

THE MASTERFUL DISASSEMBLER

Like Dis•Doc Professional, The Masterful Disassembler lets you work on your disassembly listing on the fly rather than waiting for a batch processor to finish. This is a proven product: the author of The Masterful Disassembler has used it (in various incarnations) to completely disassemble versions of CP/M and Turbo Pascal. In fact, the current version comes with a file that provides many of the labels and data definitions for Turbo Pascal 5.0. The package also includes UPACKEXE, which unpacks EXE files but not multi-segment COM files.

The Masterful Disassembler is lean and mean. It doesn't have some of the user-interface niceties found in Dis•Doc, such as mouse support. It eschews color altogether, displaying output in high-contrast black and white (see Figure 2). There's no way to highlight a

FIGURE 1: Dis•Doc's screen window

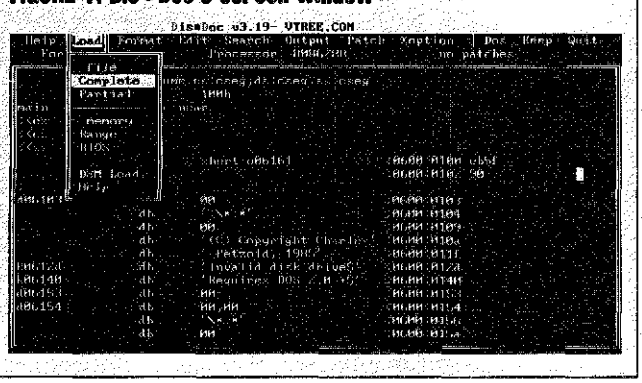


FIGURE 2: The Masterful Disassembler screen

```
00100:EB5F90      VTREE   JMP   ENTRY          ;
00103:80305C2A2E2A L0103H DB    0,'\*.*',0,'(';
0010B:432920436F70        DB    '(C) Copyright ';
00118:456861726C65        DB    'Charles Petzo';
00125:6C6420C28139        DB    'ld, 1985'     ;
0012D:496E76616C69 L012DH DB    'Invalid disk ';
0013A:647269766524        DB    'drive$'       ;
00140:5265571756972 L0140H DB   'Requires DOS ';
0014D:322E30202B24        DB    '2.0 +$'       ;
00153:00          L0153H  DB    0              ;
00154:0000        L0154H  DB    0,0            ;
00156:5C2A2E2A00  L0156H  DB    '\*.*',0       ;
0015B:0601        L015BH  DB    6,1            ;
0015D:3C03        L015DH  DB    3CH,3          ;
0015F:0000        L015FH  DB    0,0            ;
00161:3CFF        ENTRY   CMP   AL,0FFH        ;
00163:750A                JNZ   L016FH         ;
00165:BD162D01            LEA   DX,IL812DH]    ;
00169:B409CD21    L0169H  MSDOS _OUTSTR        ;Display string at (DS:DX).
0016D:CD20                INT   20H            ;Terminate program.
0016F:B430CD21    L016FH  MSDOS _GETVER        ;Get DOS version number.
00173:3C02                CMP   AL,2           ;
00175:7306                JNC   L017DH         ;
00177:BD164001            LEA   DX,IL8140H]    ;
CS:0000  F1-Help  F2-Goto  F3-SetType  F4-CheckType  F5-MakeFile  F6-Analyze
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

range of addresses to convert to code or data—you have to type in the numbers yourself. It doesn't handle addresses in segment:offset format, opting instead for 20-bit "flat" addresses.

Despite its Spartan environment, The Masterful Disassembler has a few features that Dis•Doc doesn't. It's capable of "macro decompilation," meaning that it can perform pattern matching on sequences of instructions and show them as macro invocations in the listing. Although this doesn't seem important for small programs, it's great in large ones where you don't want your listings to be any longer than necessary. It even does "macro decompilation" on DOS calls, translating them into readable macros.

DISASSEMBLE VIRUSES

Clark Calkins, the author of The Masterful Disassembler, has come up with a novel use for this facility: scanning disassembled programs for computer viruses. The disassembler comes with a disk of macros that match the "signatures" of some common viruses; if one is present, it will show up clearly in the on-screen listing.

The Masterful Disassembler's performance was similar to Dis•Doc's on most programs. In some cases it benefited from brute-force heuristics, such as intentionally not recognizing the instruction **mov [bx+si], al** (a feature that had to be turned off for TORTURE). In others, it showed a confusing display during upward scrolling. (This problem, documented in the manual, has to do with the fact that it's disassembling backward as it goes.) Its multipass data-finding algorithm was among the best at ferreting out data areas.

The manual for The Masterful Disassembler comes in two small, stapled booklets that almost fold flat if you force them. Still, the manual is one of the highlights of this product. The author describes not only his product but also the pitfalls and secrets of disassembly. This section, which by itself may be worth the price of the product, includes several interesting "war stories" about deciphering obscure code. These make excellent reading for those interested in becoming—well, masterful disassemblers.

Although Sourcer was the clear winner in the batch-mode category, your choice of interactive disassemblers will probably be a matter of personal preference. I'd recommend The Masterful Disassembler if you're a serious hacker who needs macro decompilation, isn't worried about mice and such, or is interested in purchasing label files that let you look inside products such as Turbo Pascal. Dis•Doc might be better if you need the patch feature, if you want to disassemble your BIOS easily, or if you want fancier display options such as color and varying on-screen formats.

NOOOOOO DISASSEMBLE!

In the movie *Short Circuit*, Number 5, a sapient robot, trundles about shouting "Noooooo disassemble!" (which is, of course, precisely what the bad guys want to do to him). My philosophy is similar to Number 5's: I believe that a disassembler is a tool that should be reserved for extreme circumstances, such as lost source code and emergency bug fixes. In most other cases, it will be less expensive, less time-consuming, and much less stressful to rewrite a program from scratch.

If you absolutely must disassemble, I hope the tips I've presented here will be helpful as you approach this formidable task.

VENDOR INFORMATION

The Masterful Disassembler
C.C. Software
1907 Alvarado Avenue
Walnut Creek, CA 94596
(415) 939-8153
Circle 223 on Reader Service Card

Dis•Doc Professional
RJ Swantek Inc.
P.O. Box 1032
Hartford, CT 06111
(203) 953-0236
Circle 224 on Reader Service Card

DISnDATa
Pro/Am Software
220 Cardigan Road
Centerville, OH 45459
Circle 225 on Reader Service Card

Sourcer and Unpacker
V Communications Inc.
4320 Stevens Creek Blvd, Suite 275
San Jose, CA 95129
(408) 296-4224
Circle 226 on Reader Service Card

OBFUSCATING YOUR CODE

If you want to keep your own code safe from prying eyes, here are some tactics to prevent all but the most diligent spies from disassembling your program.

Use self-modifying code, but sparingly; all it takes is one or two well-placed modifications.

Jump into the middle of instructions. This is dangerous, but you can do it if you understand the binary coding of the instruction set.

Fill uninitialized data areas with random code. This confuses readers and creates incorrect references to other parts of the program. When the disassembler checks these references, it will fail to disassemble the code properly.

Do address calculations in registers rather than by using index registers. Indexed addressing modes are easy to identify, but few disassemblers can figure out how you do math.

Use instructions for unusual purposes. How about a **ret** or an **iret** as a far jump (after pushing the target on the stack)?

Use rare instructions. Use **mov [bx+si],al** or **xlat** for some real purpose in your code. Many disassemblers turn these instructions off.

Hide your text. Encrypt, **xor** with a mask, or do something else to make the words unreadable.

Place data after conditional jumps that always succeed. The disassembler may treat it as code.

Use instructions that will cause problems for debuggers. Can you do something useful with INT 3?

Use instructions that are specific to the V20, 80186, 80286, or 80386. Use the coprocessor for integer math if it's available. Many disassemblers do not recognize these instructions unless the right option is turned on.

Scramble your jump tables. Use a simple mathematical operation on jump addresses that can be undone at runtime. Permute the bytes.

Intersperse jump vectors with other data in tables. Most disassemblers are terrible at discovering the structure of a table of records.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Exploring XMS

A Pascal utility to explore and map XMS high-memory drivers

By Kim Kokkonen

With the advent of the 286 microprocessor, PC systems received access to extended memory. (In PC terms, extended memory is above the 1-megabyte physical address boundary and is not normally accessible by DOS.) Specialized programs such as RAM disks and disk caches immediately took advantage of extended memory for their own large data structures. Unfortunately, there was no standard for managing extended memory's use. Conflicts were frequent because several programs attempted to use extended memory at the same time.

Microsoft released the HIMEM.SYS device driver in 1988. This provided a standard means of allocating and accessing extended memory. Its services are similar to those of the more familiar EMS (Expanded Memory Specification) standard developed by Lotus, Intel, and Microsoft. The interface defined by HIMEM.SYS is called XMS—eXtended Memory Specification. Other commonly used memory managers (for example, 386MAX and QEMM386) have started supporting the XMS specification, and many programs that use extended memory opt to make XMS calls rather than fighting it out for address space.

Kim Kokkonen is founder of TurboPower Software and has written many public-domain utilities used with Turbo Pascal. Kim can be reached at (408) 438-8608.

Although the 16-megabyte extended-memory addressing limit once seemed an infinite horizon, it's now becoming pretty easy to use it all up. For one thing, we have TSRs that can use EMS or XMS for swapping space. We also have RAM disks, disk caches, print buffers, and now Windows 3.0 to devour extended memory. With all of this activity, clearly it would be useful to have a utility to show how XMS memory has been allocated.

LISTING ALLOCATED BLOCKS

Unfortunately, XMS does not define a service for listing allocated blocks of extended memory. However, its **GetHandleInfo** function will return information about an extended memory block once its handle is known. Hence, one way to dump a list of the allocated blocks is to try every possible handle number (0 . . . 65535) and report information about the handle if the XMS manager doesn't return an error code. This method is slow; it takes about ten seconds to scan all the handles on a 25MHz 386 machine.

After one studies the handle-number sequence used by the XMS driver, a faster method becomes apparent. It would seem that the first handle returned by the allocation function would be one, the next two, and so on. In fact, this *is* the sequence that is used by 386MAX and QEMM386. However, Microsoft chose to be creative, and the HIMEM.SYS driver's first handle is something like 32976, with each successive handle increasing by six.

XMSDUMP.PAS demonstrates some basic functions for accessing XMS memory as well as a faster method of finding handles that are in use. The faster method allocates two dummy handles in order to get a starting point and a handle delta; then it computes the expected handle sequence and reports on those. If less than 2K of extended memory is free, XMSDUMP allocates zero-size chunks of extended memory for the dummy handles. Experimentation indicates that allocating a zero-size block will return a valid handle number; however, for some drivers the **GetHandleInfo** call appears to work intermittently for that handle. Good luck, and happy hunting. PJ

REFERENCES

- Ray Duncan's "Power Programming" column in *PC Magazine*, June 27, 1989. This is part of a series and describes XMS in detail.
- *Extending DOS*, by Ray Duncan et al. (Addison-Wesley, 1989).
- The original HIMEM.SYS driver includes a text file that describes the XMS interface. This file can be downloaded from a variety of BBSs, including CompuServe.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LISTING 1

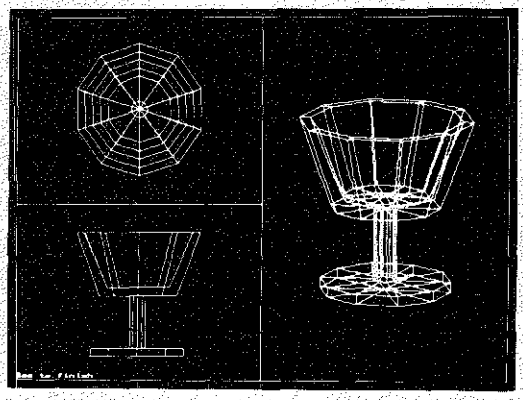
```
program XmsDump;

const
  ExhaustiveXms : Boolean = False;  {True to scan all XMS blocks}
var
  XmsControl    : Pointer;          {Pointer to XMS control procedure}

  function XmsInstalledPrim : Boolean;
    {-Returns True if an XMS memory manager is installed}
  inline(
    $B8/$00/$43/      {  MOV    AX,$4300          ; XMS Install funct}
    $CD/$2F/          {  INT    $2F               ; DOS Multiplex Int}
    $3C/$80/          {  CMP    AL,$80            ; is it there?}
    $75/$04/          {  JNE    NoXmsDriver}
    $B0/$01/          {  MOV    AL,1              ; return True}
    $EB/$02/          {  JMP    SHORT XIExit}
                      {NoXmsDriver:}
    $30/$C0);         {  XOR    AL,AL             ; return False}
                      {XIExit:}

  function XmsControlAddr : Pointer;
    {-Return address of XMS control function}
  inline(
    $B8/$10/$43/      {MOV    AX,$4310            ; XMS control func addr}
    $CD/$2F/          {INT    $2F}
    $89/$D8/          {MOV    AX,BX               ; ptr in ES:BX to DX:AX}
    $8C/$C2);         {MOV    DX,ES}

  function QueryFreeExtMem(var TotalFree, LargestBlock : Word) : Byte;
    {-Return total free and largest free block of XMS}
  var
    ErrorCode : Byte;
  begin
    inline(
      $B4/$08/            {  MOV    AH,$08   ;Query free ext memory}
      $FF/$1E/>XmsControl/{  CALL   DWORD PTR [>XmsControl]}
      $09/$C0/            {  OR     AX,AX}
      $74/$10/            {  JZ     SetError}
      $30/$DB/            {  XOR    BL,BL}
      $C4/$BE/>TotalFree/ {  LES    DI,>TotalFree[BP]}
      $26/                {ES:}
      $89/$15/            {  MOV    [DI],DX}
      $C4/$BE/>LargestBlock/{ LES   DI,>LargestBlock[BP]}
      $26/                {ES:}
      $89/$05/            {  MOV    [DI],AX}
                          {SetError:}
      $88/$5E/<ErrorCode);{  MOV    <ErrorCode[BP],BL}
    QueryFreeExtMem := ErrorCode;
  end;

  function GetHandleInfo(XmsHandle : Word;
                         var LockCount    : Byte;
                         var HandlesLeft  : Byte;
                         var BlockSizeInK : Word) : Byte;
    {-Return information about specified XMS handle}
  var
    ErrorCode : Byte;
```

continued

THE ULTIMATE CAD/CAM ENGINE

TurboGeometry Library 3.0!

Helps You build it right THE FIRST TIME!



Over 300 2D & 3D device independent geometric routines. Surfacing; Spherical & Solid Geometry; Tangents; Curves; Hidden Line; Polygons; Perspectives; 2D & 3D Clipping; Link List Library & more. In C or Pascal, w/source. 400 pg manual. $199.95, Foreign $225.00. Prices incl S&H. 30 day guarantee. VISA, MC, PO, Chk.

2116 E. Arapaho Rd.
Suite 487
Richardson, TX 75081
USA

Disk Software, Inc.

FAX or Phone
214-423-7288
Toll Free
1-800-635-7760

Circle 42 on Reader Service Card

Break Free !!!

Before

Save your Time, Money, and Sanity with our tool series for Windows 3.0 and OS/2.

	DLL	Source
SoftFIELDS™ Robust replacement Dialog Controls including: date, time, phone, money ssn, virtual multicolumn listbox, button, icon, scroll bar, user defined	149	449
SoftDDE™ Effortlessly add DDE support to your application. A clean and elegant interface with Excel, Wingz, Toolbook, Ami, and other major Windows Apps.	149	449
SoftPAGETOOL™ The foundation of every interactive graphical design application. Create Page editors similar to CorelDRAW and Toolbook. All Required Toolboxes are included.	249	749
Soft2GUI™ Complete character mode GUI for the second (mono) monitor under Windows 3.0. Redirect input between monitors. A great way to display real time data.	149	449
SoftDPMI™ No need for Assembly. Complete DPMI API in a callable DLL. Now you can access DPMI from Wingz or Toolbook Scripts.	99	199

Call TODAY for complete list of Value Added Software Modules
Supports MSC, Turbo C++, ToolBook and Windows SDK Dialog Editor

SoftWorks International

P. O. Box 8628
Atlanta, Ga 30306

To Order Call:
(404) 876-6115

After

Call about Custom Device Driver Development Services

Circle 43 on Reader Service Card

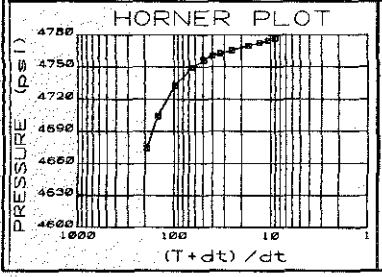
INGRAF 5.00

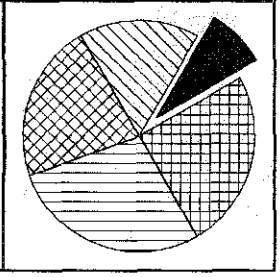
A Multi-Device Graphics Library for Scientific, Engineering and Business applications. Supports Video(CGA, EGA, VGA & Hercules), Printers (EPSON, Okidata, HP LaserJet, HP Paintjet & Postscript) and Pen Plotters(HPGL and DM/PL).

C, FORTRAN-77, PASCAL, MS Professional BASIC and QuickBASIC bindings.

FULL SOURCE CODE - NO ROYALTIES - $350.00

Sutrasoft
10506 Permian Dr.
P.O. Box 1733
Sugar Land
TX 77487-1733
U.S.A.
Tel: 713-491-2088
Fax: 713-240-6883





Circle 41 on Reader Service Card

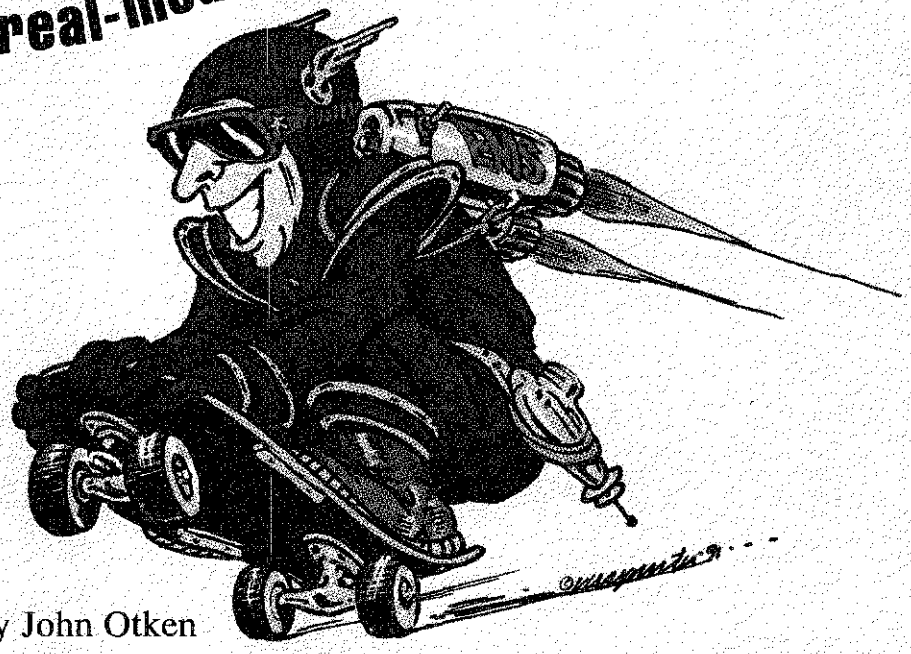
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
begin
  inline(
    $B4/$0E/                { MOV   AH,$0E   ;Get EMB Handle Info}
    $8B/$96/>XmsHandle/     { MOV   DX,>XmsHandle[BP]}
    $FF/$1E/>XmsControl/    { CALL  DWORD PTR [>XmsControl]}
    $A9/$01/$00/            { TEST  AX,1}
    $74/$17/                { JZ    SetError}
    $C4/$BE/>LockCount/     { LES   DI,>LockCount[BP]}
    $26/                   {ES:}
    $88/$3D/                { MOV   BYTE PTR [DI],BH}
    $C4/$BE/>HandlesLeft/   { LES   DI,>HandlesLeft[BP]}
    $26/                   {ES:}
    $88/$1D/                { MOV   BYTE PTR [DI],BL}
    $C4/$BE/>BlockSizeInK/  { LES   DI,>BlockSizeInK[BP]}
    $26/                   {ES:}
    $89/$15/                { MOV   [DI],DX}
    $30/$DB/                { XOR   BL,BL}
                           {SetError:}
    $88/$5E/<ErrorCode);    { MOV   <ErrorCode[BP],BL}
  GetHandleInfo := ErrorCode;
end;

function AllocateExtMem(SizeInK : Word; var XmsHandle : Word) : Byte;
  {-Allocate a block of extended memory}
var
  ErrorCode : Byte;
begin
  inline(
    $B4/$09/                { MOV   AH,$09   ;XMS Alloc ext memory block}
    $8B/$96/>SizeInK/       { MOV   DX,>SizeInK[BP]}
    $FF/$1E/>XmsControl/    { CALL  DWORD PTR [>XmsControl]}
    $A9/$01/$00/            { TEST  AX,1}
    $74/$09/                { JZ    SetError}
    $30/$DB/                { XOR   BL,BL}
    $C4/$BE/>XmsHandle/     { LES   DI,>XmsHandle[BP]}
    $26/                   {ES:}
    $89/$15/                { MOV   [DI],DX  ;return XMS handle}
                           {SetError:}
    $88/$5E/<ErrorCode);    { MOV   <ErrorCode[BP],BL}
  AllocateExtMem := ErrorCode;
end;

function FreeExtMem(XmsHandle : Word) : Byte;
  {-Free a block of extended memory given its handle}
var
  ErrorCode : Byte;
begin
  inline(
    $B4/$0A/                { MOV   AH,$0A   ;XMS Free ext memory block}
    $8B/$96/>XmsHandle/     { MOV   DX,>XmsHandle[BP]}
    $FF/$1E/>XmsControl/    { CALL  DWORD PTR [>XmsControl]}
    $A9/$01/$00/            { TEST  AX,1}
    $74/$02/                { JZ    SetError}
    $30/$DB/                { XOR   BL,BL}
                           {SetError:}
    $88/$5E/<ErrorCode);    { MOV   <ErrorCode[BP],BL}
  FreeExtMem := ErrorCode;
end;

procedure ShowTheXmsMemory;
  {-Report on allocated extended memory}
label
  ExitPoint;
var
  H0 : Word;
  H1 : Word;
  H : Word;
  Delta : Integer;
  HNum : Word;
  HMem : Word;
  FMem : Word;
  FMax : Word;
  Total : Word;
  Status : Byte;
  LockCount : Byte;
  HandlesLeft : Byte;
  Done : Boolean;
begin
  if XmsInstalledPrim then
    XmsControl := XmsControlAddr
  else begin
    WriteLn('No XMS driver installed');
    Exit;
  end;
  Status := QueryFreeExtMem(FMem, FMax);
  if Status = $A0 then begin
    {All XMS has been allocated}
    FMem := 0;
    FMax := 0;
  end else if Status <> 0 then begin
    WriteLn('Error ', Status, ' accessing XMS');
    Exit;
  end;

  WriteLn('block   bytes   (XMS Memory)');
  WriteLn('-----  -------');

  {Total will count total XMS memory}
  Total := 0;
  {HNum will list the XMS handles in sequential order}
  HNum := 0;

  if ExhaustiveXms then begin
    {Search all 64K XMS handles for valid ones}
    for H := 0 to 65535 do begin
      Status := GetHandleInfo(H, LockCount, HandlesLeft, HMem);
      if Status = 0 then begin
        WriteLn(HNum:5, '  ', LongInt(1024)*HMem:7);
        inc(Total, HMem);
        inc(HNum);
      end;
    end;

  end else begin
    {Heuristic algorithm to report used handles quickly}

    {Allocate two dummy handles}
    if FMem > 1 then
      HMem := 1
    else
      HMem := 0;
    Status := AllocateExtMem(HMem, H0);
    if Status <> 0 then
      goto ExitPoint;
    Status := AllocateExtMem(HMem, H1);
    if Status <> 0 then begin
      {Deallocate dummy handle}
      Status := FreeExtMem(H0);
      goto ExitPoint;
    end;
    Delta := H1-H0;
    {Deallocate one dummy}
    Status := FreeExtMem(H1);

    {Trace back through valid handles}
    H := H0;
    repeat
      Status := GetHandleInfo(H, LockCount, HandlesLeft, HMem);
      Done := (Status <> 0);
      if not Done then
        dec(H, Delta);
    until Done;

    {Go forward again through valid handles, reporting them}
    inc(H, Delta);
    while H <> H0 do begin
      Status := GetHandleInfo(H, LockCount, HandlesLeft, HMem);
      if Status = 0 then begin
        WriteLn(HNum:5, '  ', LongInt(1024)*HMem:7);
        inc(Total, HMem);
        inc(HNum);
      end;
      inc(H, Delta);
    end;

    {Deallocate dummy handle}
    Status := FreeExtMem(H0);
  end;

  inc(Total, FMem);

ExitPoint:
  WriteLn('free  ', LongInt(1024)*FMem:7);
  if Total <> 0 then
    WriteLn('total ', LongInt(1024)*Total:7);
end;

begin
  ShowTheXmsMemory;
end.
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Using EMS for fast file I/O

More about real-mode virtual memory



By John Otken

My *PJ* 9.1 article on real-mode virtual memory featured a high-performance file system. This article will continue with the code for that system, focusing on the Expanded Memory Specification (EMS). EMS defines a method with which DOS programs can access large amounts of memory outside the normal limit of 640K. A program cannot assume that a PC supports EMS until it checks for a driver. In an assembly language program, the check is easy: grab interrupt vector 67h, use 000Ah instead of the offset from the interrupt vector, and then look for the eight-character tag EMMXXXX0.

If the tag is there, invoke interrupt 67h with AH=46h. If AH equals zero on return from the interrupt, then register AL holds the EMS driver version number. The two common versions are 32h and 40h. For an example, see **get_ems_version** in Listing 1.

PAGES OF MEMORIES

EMS 3.2 organizes memory into 16K pages. A program can allocate any number of pages from the available pool and map any of these pages to one of four regions of a 64K frame in memory between 640K and one megabyte. With hardware support, this system provides quick access to expanded memory. However, it can be difficult to manage for certain kinds of programs. For example, a sorting algorithm may have to call the EMS driver before each access to a record.

TSRs also have problems with EMS 3.2. They have to save and restore the state of the page frame, which can take too long for real-time interrupt processing. They must also check for conflicts with the page frame: for example, some disk caches break when a program reads a file into an EMS page.

EMS 4.0 adds a much better interface to expanded memory and also supports the old 3.2 functions. A program still allocates memory in 16K pages under 4.0, but it can transfer any range of memory, from one byte to one megabyte. EMS 4.0 memory appears as a linear space. A program can transfer or exchange either between or within main and expanded memory.

The page frame is obsolete under EMS 4.0, so TSRs can access small sections of EMS quickly and without disturbing other programs.

EMS AND VIRTUAL MEMORY

Because EMS is fast, it is a key component of the virtual memory system. Disk swapping is supported only so that worst-case programs can run; after all, it would be a shame to abort a program that needed only one more block.

John Otken is president of Soft Advances, where he hacks assembly language into debuggers and other software for embedded systems. Contact him at P.O. Box 184, Orinda, CA 94563.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LISTING 1

```
        title   expanded memory interface
        include asm.inc

        public  ems_exchange
        public  ems_in
        public  ems_out

EMS_MAX         equ     1024    ; (must be multiple of 8)
NULL_EMS        equ     0

move_source_dest_struct  struc   ; from official LIM EMS 4.0 Spec - p. 3-87
  region_length                  dd ?
  source_memory_type             db ?
  source_handle                  dw ?
  source_initial_offset          dw ?
  source_initial_seg_page        dw ?
  dest_memory_type               db ?
  dest_handle                    dw ?
  dest_initial_offset            dw ?
  dest_initial_seg_page          dw ?
move_source_dest_struct  ends


XIB     segment word public 'DATA'      ; enable exit intercept during
XIB     ends                            ; initialization
XI      segment word public 'DATA'
        dw      enable_exit_intercept
XI      ends
XIE     segment word public 'DATA'
XIE     ends

XCB     segment word public 'DATA'      ; always release EMS on termination
XCB     ends
XC      segment word public 'DATA'
        dw      ems_close
XC      ends
XCE     segment word public 'DATA'
XCE     ends


        .data?
ems_struc       move_source_dest_struct <>

ems_count       dw      ?               ; number of allocated pages
ems_handle      dw      ?

ems_ecount      dw      ?
ems_icount      dw      ?
ems_ocount      dw      ?

ems_version     db      ?
ems_flag db     ?

ems_bits db     EMS_MAX/8 dup(?)


        .const
ertx_ems_index  db      'Bad EMS index',0
ertx_ems_error  db      'Error in EMS',0


        .code
        extn    set_strerror,get_vector,enable_exit_intercept


;;      do ems function
;
;       exit    Cf      if error (sets error string)
;
do_ems_function proc
        int     67h
        or      ah,ah
        jnz     def1                    ;  if EMS error
        ret

def1:   lea     ax,ertx_ems_error       ; *Error in EMS*
        jmp     set_strerror
do_ems_function endp


;;      do ems raw
;
;       exit    Cf      if error (doesn't set error string)

;
do_ems_raw proc
        int     67h
        or      ah,ah
        jnz     der1                    ;  if EMS error
        ret
der1:   stc
        ret
do_ems_raw endp


;;      ems close
;
;       uses    AX,DX
;
ems_close proc
        movx    dx,NULL_EMS
        xchg    dx,ems_handle[bp]
        cmpx    dx,NULL_EMS
        je      ecl1                    ;\ if EMS not in use

        mov     ah,45h                  ; deallocate pages
        call    do_ems_function
ecl1:   ret
ems_close endp


;;      ems exchange
;
;       entry   AX      page index (1..n)
;               ES:DI   destination
;       exit    Cf      if bad block or EMS error
;       uses    AX
;
ems_exchange proc
        pushm   si,ds                   ; adjust and set EMS page index
        dec     ax
        mov     ems_struc.dest_initial_seg_page[bp],ax

        call    read_move_struc
        jc      eex1                    ;  if no EMS
        mov     dest_handle[si],ax

        movx    ax,0                    ; fill in move structure
        mov     source_memory_type[si],al
        mov     source_handle[si],ax
        mov     source_initial_offset[si],di
        mov     source_initial_seg_page[si],es
        mov     dest_memory_type[si],1
        mov     dest_initial_offset[si],ax

        mov     ax,5701h                ; exchange RAM and EMS
        call    do_ems_function
        inc     ems_ecount[bp]

eex1:   popm    ds,si
        ret
ems_exchange endp


;;      ems in
;
;       entry   AX      block index (1..n)
;               ES:DI   destination (no transfer if NULL)
;       exit    Cf      if bad block or EMS error
;       uses    AX
;
ems_in proc
        pushm   cx,si,ds
        dec     ax                      ; clear index bit
        js      ein2                    ;  if bad swap index (cannot be -)
        mov     si,ax
        mov     cl,3                    ;  (divide bit number by 8 to se-
lect
        shr     si,cl                   ;   byte)

        mov     cl,al                   ;  (use the 3 least significant
bits
        and     cl,7                    ;   to rotate a mask)
        mov     ch,10000000b
        ror     ch,cl
        test    ems_bits[bp+si],ch
        jz      ein2                    ;  if bad EMS index
        xor     ems_bits[bp+si],ch      ;  (Cf=0)
```

continued

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

An EMS 4.0 driver keeps the code in EMS.ASM (Listing 1) as simple as possible; older 3.2 drivers are ignored. A commercial version of the virtual memory system should support older EMS drivers, but this is not the case here.

EMS.ASM interfaces to expanded memory through three procedures: **ems_ exchange** exchanges blocks in EMS and main memory; **ems_in** moves a block from EMS to main memory; and **ems_out** moves a block from main memory to EMS.

Ems_in and **ems_out** maintain a bitmap of all the currently allocated EMS blocks. **Ems_in** clears a block's bit after a transfer. **Ems_out** searches for the first free block, marks it, and then transfers. If **ems_out** cannot find a free block, it tries to allocate another block from the system.

All three procedures use the same register conventions. AX holds a nonzero EMS block index, and ES:DI holds a pointer to a block in main memory. EMS.ASM relies on the start-up code to intercept program termination. This guarantees that a program will always free its expanded memory. Otherwise, EMS.ASM is self-contained.

A good reference for EMS is the *DOS Programmer's Reference* (2nd Edition) by Que. Functions are marked as 3.2 or 4.0, so a programmer can support drivers compatible with either specification.

TIGHT MODEL PROGRAMMING

Let's look at the revised coding conventions. *Tight model* is for programming assembly language. It is a superset of Microsoft's *small model*, with more details about program appearance.

Most of the tight model programs differ from small model ones in two ways. In small model, register DS is restricted to DGROUP. In tight model, DS can point to any segment. Unrestricting DS makes it easier to access large data spaces. To reflect this change, a tight model program must have an **assume ds:nothing** statement.

The two model programs use register BP differently. In a small model program, BP usually points to a stack frame for local variables and parameters. In tight model, BP is always zero.

When an instruction indexes with BP, the CPU adds BP to the effective address and uses SS, the stack segment

New Version 3.0

Commenting Disassembler!

SOURCER™ 486

- **SEE HOW PROGRAMS WORK**
- **EASILY MODIFY PROGRAMS**

SOURCER™ creates detailed commented source code and listings from memory and executable files. Built in data analyzer and simulator resolves data across multiple segments and provides detailed comments on interrupts and subfunctions, I/O ports and much more. Determines necessary assembler directives for reassembly. Includes a definition file facility to include your own remarks and descriptive labels, force data types, and more. Complete support for 8088/87 through 80386/387, 80486, and V20/V30 instruction sets. We welcome comparisons with any other product, because no product comes close to the ease of use and output clarity of SOURCER.

Sourcer is the best disassembler we've ever seen!
—PC Magazine, January 17, 1989, page 101

SAMPLE OUTPUT

Fully automatic

Program header

Assembler directives

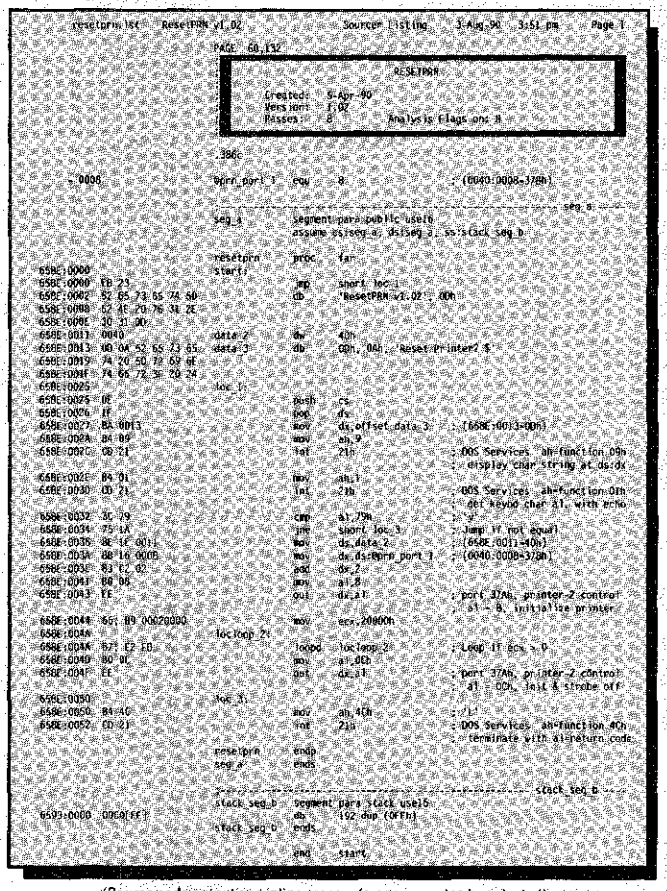
Determines data areas and type

Detailed comments

Simulator follows segment changes

80386 and 80486 support

Easy to read format



(Source code output and inline cross reference can also be selected)

BIOS SOURCE

- **CHANGE AND ADD FEATURES**
- **CLARIFY INTERFACES**

for PS/2, AT, XT, PC, and Clones

The BIOS Pre-Processor™ with SOURCER provides the first means to obtain accurate legal source listings for any BIOS! Identifies entry points with full explanations. Resolves PS/2's multiple jumps for improved clarity. Provides highly descriptive labels such as "video_mode" and much more. Fully automatic.

SOURCER Commenting disassembler	$129.95	**ASMtool 486™** Assembly source analyzer and flowcharter	$199.95
SOURCER with BIOS Pre-Processor	169.95	**View-It™** High speed zoomable file viewer	69.95
UNPACKER™ Unpack packed EXE files	39.95	**ASM ProPak II™** Sourcer w/BIOS, UnPacker, ASMtool and View-It	399.75

Shipping & Handling: USA $5; Canada/Mexico $10; Other $15; CA Res. add sales tax; PS/2 trademark of IBM Corp

NOT COPY PROTECTED **30-DAY MONEY-BACK GUARANTEE**

VISA **1-800-648-8266** MasterCard

V COMMUNICATIONS, INC.

4320 Stevens Creek Blvd., Suite 275, Dept. PJ8, San Jose, CA 95129 (408) 296-4224

Circle 44 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

register, instead of DS, the data segment. If BP is zero, then indexing with BP provides a "free" way for instructions to access memory via the SS register. Because SS always points to DGROUP, access to static variables in DGROUP is efficient.

Tight model uses the same segments as small model, but DGROUP must align to a paragraph boundary. Alignment is easy with full segment specifications, but programs using simplified segment directives require a dummy segment definition:

```
NULL segment para public 'BEGDATA'
NULL ends
```

In a tight model program, program layout is conventional. The general order of program elements is title, include files, entry points, equates, structure definitions, data, and code.

"Entry points" are PUBLIC statements that declare the interface to the module. Do not mix general public declarations with entry points. Alphabetizing procedures in the code section makes procedures easier to find and guarantees the same order no matter who is programming.

NAMES AND COMMENTS

Tight model specifies a method for naming procedures in a program. When possible, borrow function names from well-known sources such as the standard C library. Otherwise, use three-word names with a verb_adjective_noun form. An example is **read_next_char.** For data, use two-word names of the form **word_count**.

If a procedure requires local labels, create them from the procedure name. With a three-word procedure name, use the three initial letters plus a number. For example, **read_next_char** would use **rnc1** and **rnc2** for local labels.

A multiline procedure comment precedes each procedure in a tight model program. The comment serves to document the procedure name, the entry and exit conditions of the registers, and the registers that have been bashed by the routine. A sample of a multiline procedure comment follows; see Listing 1 for more.

```
;;  ems in
;
;   entry   AX      block index (1..n)
;           ES:DI   destination
;                     (no transfer if NULL)
;   exit    Cf      if bad block or EMS error
;   uses    AX
```

Assembly language programs typically fall into small groups of instructions that perform simple functions. When the function of one of these small groups isn't immediately obvious, describe the function with an in-line comment on the first instruction. The following is a block comment:

```
        mov     ah,46h              ; get EMS version
        call    do_ems_function
```

Instructions in a small groups such as these seldom need their own in-line comments. Conditional branches frequently do need a comment to describe their condition. The comments should

```
        mov     cx,es                   ; don't read EMS file if NULL output
        jcxz    ein1                    ;  if NULL destination (Cf=0)

                                        ; set struct for move memory region
        mov     ems_struc.source_initial_seg_page[bp],ax

        call    read_move_struc
        jc      ein1                    ;  if no EMS
        mov     source_handle[si],ax

        movx    ax,0
        mov     source_memory_type[si],1
        mov     source_initial_offset[si],ax
        mov     dest_memory_type[si],al
        mov     dest_handle[si],ax
        mov     dest_initial_offset[si],di
        mov     dest_initial_seg_page[si],es

        mov     ax,5700h                ; move from EMS to RAM
        call    do_ems_function
        inc     ems_icount[bp]

ein1:   popm    ds,si,cx
        ret

ein2:   lea     ax,ertx_ems_index       ; *Bad EMS index*
        call    set_strerror
        jmp     ein1
ems_in endp


;;      ems out
;
;       entry   ES:DI   source
;       exit    AX      block index (1..n)
;               Cf      if EMS full or EMS error
;
ems_out proc
        pushm   cx,si,ds
        call    read_move_struc         ; set struct for move memory region
        jc      eou2                    ;  if no EMS
        mov     dest_handle[si],ax

        movx    ax,0
        mov     source_memory_type[si],al
        mov     source_handle[si],ax
        mov     source_initial_offset[si],di
        mov     source_initial_seg_page[si],es
        mov     dest_memory_type[si],1
        mov     dest_initial_offset[si],ax

        call    get_ems_index           ; select next free EMS block
        jc      eou2                    ;  if EMS bit map full
        mov     dest_initial_seg_page[si],ax
        mov     cx,ax

        cmp     ax,ems_count[bp]        ; check EMS allocation
        jb      eou1                    ;  if enough EMS pages allocated
        call    realloc_more_ems        ;  else get more ESM from driver
        jc      eou2                    ;   if EMS full

eou1:   mov     ax,5700h                ; move from EMS to RAM
        call    do_ems_function
        inc     ems_ocount[bp]

        mov     ax,cx                   ;  return EMS index (1..MAX)
        inc     ax

eou2:   popm    ds,si,cx
        ret
ems_out endp


;;      get ems index
;
;       exit    AX      ems index (0..MAX-1)
;               Cf      if no (more) EMS
;       uses    CX
;
get_ems_index proc
        push    si
        mov     al,1
        mov     cx,EMS_MAX
        lea     si,ems_bits-1

        even                            ; search ems bits for first free
index
gei1:   ror     al,1
```

continued

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

be of the "if (condition)" form, where (condition) describes the condition of the branch.

```
cmp  ax,NULL_EMS  ; reject a zero EMS handle
                  ;  (Cf=0)
je   rms3         ;  if bad handle
```

Other instructions in a group may need in-line comments. It's a good idea to make these distinct from block comments by indenting an extra space or by enclosing them in parentheses, as in the following example:

```
dec  ax      ; clear index bit
mov  si,ax
mov  cl,3    ;  (divide bit number by 8
shr  si,cl   ;   to select byte)
```

Tight model programs avoid macros because they make a program difficult to read and maintain. Exceptions are those specifically designed to improve readability. The tight model **include** file defines some "good" macros.

Conditional assembly also makes a program difficult to read and maintain and should be avoided; it is better to use extra instructions than to use conditional assembly. For example:

```
ifdef OEM_VERSION        ; this is bad
call  oem_preset
endif

test  flags,OEM_VERSION  ; this is less
jz    mail               ;  efficient, but
call  oem_preset         ;  better
```

Tight model programs use an include file, ASM.INC. It selects the small memory model and ASSUMEs register DS to nothing. It equates some symbols and defines some macros. Three of the equates (*bptr*, *wptr*, and *dptr*) shorten pointer expressions:

```
bptr  equ  byte ptr
mov   bptr [si],'@'
```

The **movx** macro provides a more readable way to move between segment registers or to zero a register; compare:

```
movx  es,ds
movx  ax,0

push  ds
pop   es
xor   ax,ax
```

ASM.INC also includes six macros for MASM and TASM that duplicate the built-in statements from OPTASM. The **pushm** and **popm** macros are similar to **push** and **pop**, but they allow multiple registers in one line; **extb**, **extw**, **extd**, and **extn** are condensed forms of **extrn**. Particularly valuable for long lists of near externals is **extn**.

```
pushm ax,bx,cx,dx ; this line instead
push  ax          ;  of these
push  bx          ;  4 lines
push  cx
push  dx

extn  strlen,strcpy           ; this instead
extrn strlen:near,strcpy:near ;  of this
```

CODE, TOOLS, AND NEWS

Tight model programs use start-up code similar to that in C programs. The code initializes dynamic storage, zeros the uninitialized data segment (BSS), processes command line arguments, and calls module initializers and **main**. When **main** returns, it calls on-exit routines and returns to DOS. Two utilities make

Total control with LMI FORTH™

For Programming Professionals: an expanding family of compatible, high-performance compilers for microcomputers

For Development:
Interactive Forth-83 Interpreter/Compilers for MS-DOS, OS/2, and the 80386
- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 500 page manual written in plain English
- Support for graphics,floating point, native code generation

For Applications: Forth-83 Metacompiler
- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, 6303, 6809, 68HC11,34010, V25, RTX-2000
- No license fee or royalty for compiled applications

Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone Credit Card Orders to: (213) 306-7412
FAX: (213) 301-0761

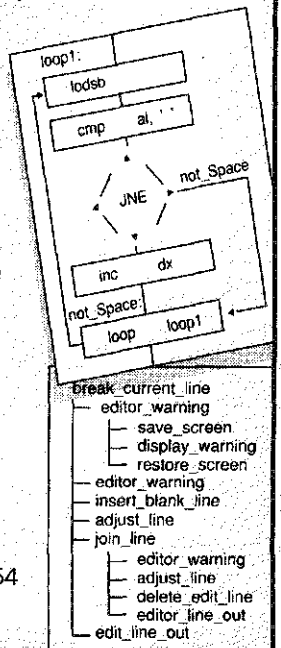
Circle 45 on Reader Service Card

ASMFLOW PROFESSIONAL

YOU CAN STEP BACK AND TAKE A LOOK AT YOUR CODE

$199.95 S/H $5.00 Outside U.S. add $10.00

FLOW CHART AND ANALYZE YOUR ASSEMBLY LANGUAGE SOURCE CODE
- Flow Charts
- Tree Diagrams
- Stack Sizing
- Register Analysis
- CPU Timing Analysis
- Procedural X-Reference
- 8088/87 to 80486
- Context-Sensitive Help
- MASM 5.1 Compatible
- Macro Expansion
- 8085, Z-80 and 8051 Versions



QUANTASM CORPORATION
19855 Stevens Creek Blvd, Suite 154
Cupertino, CA 95014
(408) 244-6826 – Free Demo

VISA • MC 30 – DAY MONEY BACK GUARANTEE

Circle 46 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ADVERTISER INDEX

Reader Service	Advertiser Name	Page Number
2	Acucobol, Inc.	1
34	Alphax Systems, Inc.	51
55	Artisoft Inc.	C3
22	Austin Code Works, The	31
110	Az-Tech Software, Inc.	94
29	Blaise Computing, Inc.	41
56	Borland International	C4
*	Borland Languages Show	57
118	Burton Systems Software	94
130	C.C. Software	95
16	Catenary Systems	22
*	Cater Software	86
*	CDP Consultants	95
127	Cnapse	95
51	Cobalt Blue	90
8	Coleman Softlabs	12
128	Device Drivers, Inc.	95
42	Disk Software, Inc.	73
*	DSE Software Publishing	94
111	EZX	94
7	Genus Microprogramming, Inc.	10
*	Gimpel Software	37
37	Goin Places Software	52
5	Greenleaf Software, Inc.	7
*	GUI Computer	37
123	Impulse Engineering	94
*	IMSL, Inc.	59
116	Information Modes	95
112	Interstitial Software Systems	94
40	Kadak Products, Ltd.	84
14	Knowledge Dynamics Corp.	19
45	Laboratory Microsystems, Inc.	79
36	Lahey Computer Systems, Inc.	52
21	Magee Enterprises, Inc.	27
*	Mark Williams Company	11
12	MetaWare Incorporated	15
35	Micro Security Systems	51
9	Midnight Engineering	37
28	Mix Software	39
32	Modular Software Systems	47
38	Nanosoft	56
125	Network Integrated Services	94
4	Nu-Mega Technologies	4
49	Object Magazine	89
*	Olivadoti Associates, Inc.	3
15	OPENetwork	19
113	Opt-Tech	95
50	Opus Software	90
121	Orion Software	95
31	OTG Systems, Inc.	45
30	PC Tech, Inc.	45
52	PC Tech, Inc.	41
39	Periscope Company, Inc., The	63
*	***PJ* Mail Order Store**	81
10	Programmer's Corner, The	14
*	Quaid Software Limited	54
46	Quantasm Corporation	79
1	Quarterdeck Office Systems	C2
3	Quarterdeck Office Systems	2
*	Que Corporation	53
47	Quibus Enterprises	86
33	Quinn-Curtis	50
19	Rational Systems, Inc.	25
6	RJSwantek & Associates	9
117	Robin Rowe Software	94
115	Ryle Design	95
48	S.L.A.M. Software	86
13	Sapiens Software Corp.	19
20	Schroff Development	26
126	Sequiter Software, Inc.	94
124	Software Mart	94
120	Software Success	95
129	Software Toolz, Inc.	95
18	Softway, Inc.	24
43	Softworks	73
17	Sub Systems	22
41	Sutrasoft	73
25	Symmetry Group, The	34
26	Tool Makers, The	36
11	Tools & Techniques, Inc.	14
114	TSR Systems	95
54	Turbopower Software	93
122	Upper Deck Systems	95
44	V Communications	77
*	Vermont Database Corp.	94
*	Watcom Products, Inc.	35
119	Wenham Software Co.	94
27	Western Wares	39
24	Winsoft	33
23	Zircel Software	32

When contacting any of the above advertisers, please tell them you saw their ad in ***Programmer's Journal*** and that you appreciate their continuing support of this magazine. The advertisers marked with an asterisk (*) prefer to be contacted directly. ***PJ***'s Advertiser Index is provided as a free service, and ***Programmer's Journal*** cannot be held responsible for omissions or errors. ***Programmer's Journal*** is not responsible for errors in advertisements.

PJ Code is available on the following bulletin board services:

Buffer Board
538 Cardinal Way
Springfield, OR 97477
(Data) 503-747-7636
The Buffer Board contains all of PJ's code from Sept/Oct 1987 issue through the current issue.

A.C.M. Phoenix Chapter
P.O. Box 47302
Phoenix, AZ 85068
(Data) 602-970-0474

Mnematics Videotex
P.O. Box 19
Sparkill, NY 10976-0019
(Voice) 800-322-3633
(Voice) 914-365-0184 in N.Y.
*Mention **PJ** when you call.*

Gilmore Systems of Southern California
5447 Indian Hills Drive
Simi Valley, CA 93063
(Voice) 805-582-1360
(Data) 805-582-9306

Ithaca Street BBS
1145 Ithaca
Boulder, CO 80303
(Voice) 303-494-8865
(Data) 303-494-8868

The Programmer's Corner
5430F Lynx Ln. #224
Columbia, MD 21044
(Data) 301-596-1180, 621-4636, or 995-3744

Lynzie's Motherboard
3960 Laurel Canyon Blvd.,
Suite 121
Studio City, CA 91614
(Voice) 213-654-0526
(Data) 213-650-0519 or 650-1233
Type PJ after your name at LOGON for access within 24 hours without verification.

PCS-49 on CompuServe
Section 10
Sponsored by:
MISOSYS, Inc.
P.O. Box 239
Sterling, VA 22170
(Voice) 703-450-4181

Generation Five Inc.
12061 Tech Road
Silver Spring, MD 20904-9998
(Voice) 301-680-3600
(Data) 301-680-9300
Accessible via PC Pursuit, Washington, D.C.
From Files submenu, select "P" for Personal Downloads.

The Silicon Zone
820 Tivoli Circle #203
Deerfield Beach, FL 33441
(Voice) 305-360-9362
(Data) 305-360-9376

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

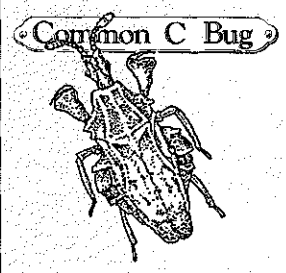
PJ ISSUE SPECIAL!

SAVE $2.00 ea.

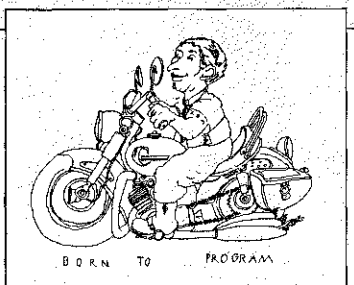
Multi-color imprints on the highest quality 100% cotton T-shirts.

SIZES: S, M, L, XL, XXL

$14.95 each — includes U.S. shipping



COMMON C BUG



BORN TO PROGRAM

ALL CODE DISKS ONLY $8.00 EACH!

For a limited time, you can save 20% on each code disk you order. Save yourself time and money—order the disks you need today!

Free U.S. shipping. Add $2.50 shipping per disk to Canada or Mexico, $5.00 shipping per disk to any other foreign destination. Offer expires 4-30-91. Not valid for PGP disk.

PARALLEL PORTS

PARALLEL PORTS

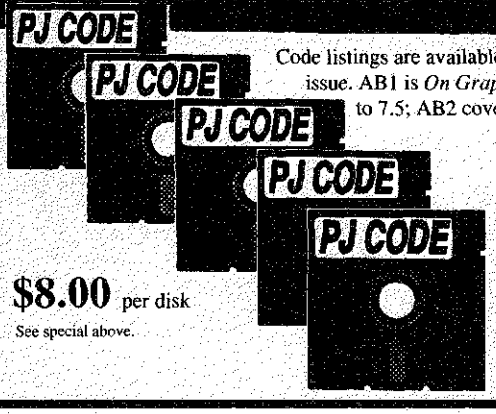
RS-232 head

RS-232 HEAD

BUGS RULE 4 color imprint on a white shirt

All mail orders must be accompanied by check or Money Order -- PREPAID ONLY.

Use the order form on the card opposite.
Order by phone with your VISA/MC.
CALL TOLL FREE
800/234-0386 ext 53



$8.00 per disk
See special above.

Code listings are available from volume 5.5 to the current issue. AB1 is *On Graphics* series code from issues 5.1 to 7.5; AB2 covers issue 7.6 to the current issue.

Diskettes are available for the following code:

AB1	6.3	7.3	8.3
AB2	6.4	7.4	8.4
5.5	6.5	7.5	8.5
5.6	6.6	7.6	8.6
6.1	7.1	8.1	9.1
6.2	7.2	8.2	9.2

360K MS-DOS disks standard; other formats are available.

$7.00 each PREPAID ONLY

A few of the Back Issues currently available

Vol. 7.6: USER INTERFACES: Windows terminal program; the CURSES C library; two video displays with Turbo Pascal; VGA and EGA split screens; memory models and static variables for assembly language; object-oriented programming to create a file viewer; large memory spaces from DOS on the 80386; QuickBASIC's string-searching algorithm; reviews of user-interface and mini-Case tools; the *System BIOS for IBM PC, XT, AT.*

Vol. 8.1: NETWORKING: Network File System; an overview of NetBIOS on the PC, from *C Programmer's Guide to NetBIOS* by W. David Schwaderer; fast circle drawing on the VGA; Dynamic Data Exchange in Windows; DESQview API features; computing time in C; a menu system with Turbo Pascal 5.5.

Vol. 8.2: GRAPHICS: Faster circles for the VGA; the TIGA interface from TI; the EGA palette; 800x600x16 Super-VGA mode programming; dynamic data exchange for communications; Manipulating large data; fast file I/O in QuickBASIC; industry news; 1989 article index.

Vol. 8.3: OPERATING ENVIRONMENTS: Undocumented DOS shell; graphics programming in windows; mousing around: messages in PM; accessing high memory; ASM:advanced segmentation; ellipses on the VGA; Up OOP alley in Pascal.

Vol. 8.4: PROGRAMMING IN THE PROTECTED MODE: Faster VGA ellipses and more; Running unprotected virtual 8086 mode; Converting to 386 protected mode; OS/2 2.0 memory model for the 386; Fortran for the 386; Using C++ features in C; Different times compensating for system clock idiosyncracies; ASM:Errors and Stream I/O; Cobol-85 nested; Deliberately breaking the rules.

Vol. 8.5: OBJECT_ORIENTED PROGRAMMING: More colors on the VGA; A virtual memory system in C/C++; Pitfalls in Turbo Pascal OOP; Writing applications that travel; Late binding; Tools for 386 DOS; Windowing text with X; Cache memories; Breaking the rules again; SPARCs plot world takeover.

Vol. 8.6: THE DATABASE CHALLENGE: VGA color paging; The database dilemma; Networking databases; B-tree indexing; Virtual memory paging; Writing an ANSI C preprocessor; Peak performance: on to the 80486; Calendar conversions; Software components; Optimization strategies; Pink noise; Oh say can you C?

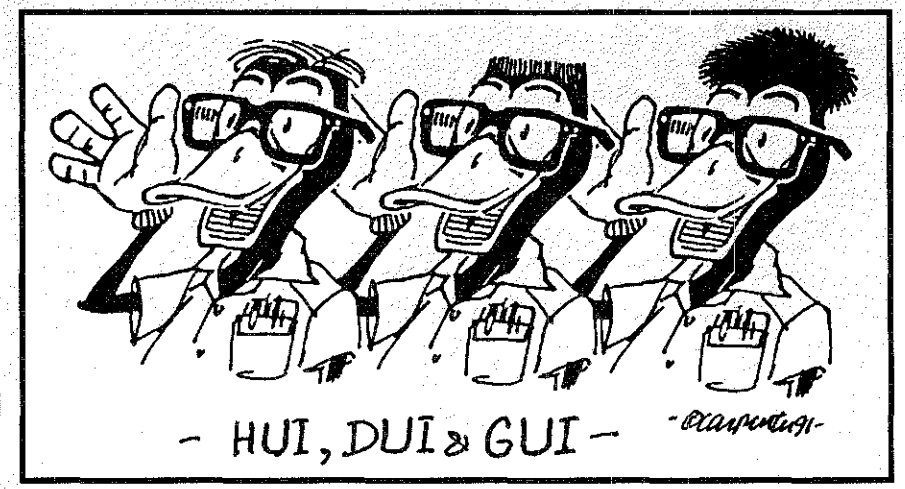
Vol. 9.1: GRAPHICS AND WINDOWING: VGA color cycling; Digital typography; Windowing for two; GUI madness; Text buffers in windows; What's on the menu; 386 DOS meets ANSI C; Optimania; Real-mode virtual memory; Enhanced keyboards; An open look at Motif; Negotiating tricks and traps; For a few good books; Goliath beats David—As usual.

10% discount on any order of 10 or more T-Shirts, Disks or Back Issues.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

tight model programming easier. ASMHDR reads one or more modules, extracts and sorts the procedure comments, and writes them to a file. LIBGEN builds command files for MAKE, LIB, OPTASM, and OPTLIB from a list of source files and helps to build a library (LIB) from a directory of source files.

The new version of MASM is in beta test and should be ready soon. If it is, my next article will compare it to the latest releases of TASM and OPTASM. The Microsoft people are sure that their new release will be as fast as TASM. At any rate, Microsoft will probably make the same claims about speed in their advertisements as Borland does. PJ




```
        adc     si,ZERO
        test    al,[bp+si]
        loopnz  gei1
        stc
        jnz     gei2                    ; if EMS full
        or      [bp+si],al

        mov     ax,EMS_MAX-1
        sub     ax,cx

gei2:   pop     si
        ret
get_ems_index endp


;;      get ems version
;
;       exit    AL      EMS version or 0 if no EMS
;       uses    SI,DS
;
get_ems_version proc
        mov     al,67h
        call    get_vector
        mov     si,0Ah
        lodsw
        cmp     ax,'ME'
        jne     gev1                    ; if no EMS
        lodsw
        cmp     ax,'XM'
        jne     gev1                    ; if no EMS
        lodsw
        cmp     ax,'XX'
        jne     gev1                    ; if no EMS
        lodsw
        cmp     ax,'0X'
        jne     gev1                    ; if no EMS

        mov     ah,46h                  ; get EMS version
        call    do_ems_function
        jnc     gev2

gev1:   movx    ax,0
        stc
gev2:   ret
get_ems_version endp


;;      read move struc
;
;       exit    AX      EMS handle
;               DS:SI   "move source dest" structure
;               Cf      if no EMS
;       note    all members set to zero except region_length
;
read_move_struc proc
        mov     ax,ems_handle[bp]
        cmpx    ax,NULL_EMS
        je      rms2                    ; if no ems

rms1:   movx    ds,DGROUP_SEGMENT
        lea     si,ems_struc
        mov     wptr region_length[si],BLOCK_SIZE
        mov     wptr region_length[si+2],ZERO
        clc
        ret

rms2:   mov     al,-1
        xchg    al,ems_flag[bp]
        add     al,-1
        jc      rms4                    ; if previous attempt failed

        call    get_ems_version
        mov     ems_version[bp],al
        cmp     al,40h
        jb      rms4                    ; if EMS missing or out of date

rms3:   mov     ah,43h                  ; get EMS handle
        pushm   bx,dx
        mov     bx,1                    ; allocate just 1 page
        call    do_ems_raw
        mov     ax,dx
        popm    dx,bx
        jc      rms4                    ; if no EMS available (probably)
        mov     ems_handle[bp],ax
        mov     ems_count[bp],1

        cmpx    ax,NULL_EMS             ; reject a zero EMS handle (Cf=0)
        je      rms3                    ;  if bad handle
        jmp     rms1

rms4:   ret
read_move_struc endp


;;      realloc more ems
;
;       exit    Cf      if no more EMS
;       uses    AX
;
realloc_more_ems proc
        pushm   bx,dx
        mov     bx,ems_count[bp]
        inc     bx
        mov     dx,ems_handle[bp]
        mov     ah,51h
        call    do_ems_raw
        jc      rme1                    ; if out of EMS
        mov     ems_count[bp],bx
rme1:   popm    dx,bx
        ret
realloc_more_ems endp

        end
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Language lore

By M. Steven Baker

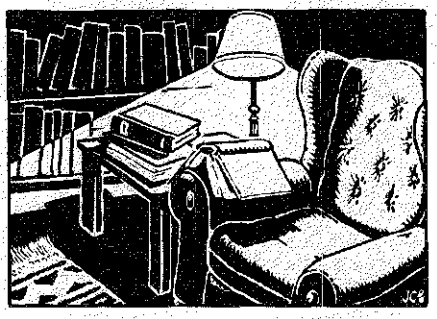


ILLUSTRATION BY JIM CARPENTER

Computer languages seem to be as addictive as coffee or cigarettes—once you become a user of one, you don't want to switch. This issue of *PJ* focuses on various languages, so I'll review books about languages. I spoke with a number of *PJ* authors, programmers, and software vendors, asking them for their favorite books in the programming language of their choice. From this long list, I culled out those I found of most interest. I'll also mention several newly revised editions on graphics and the premier book on undocumented DOS, so read on.

MY LANGUAGE IS BETTER ...

C is the computer tongue currently in vogue, so let's start with it. My pick for the single best reference book on C is Prentice Hall's ***C: A Reference Manual***, by Samuel Harbison and Guy Steele. This third edition includes both ANSI and traditional (K&R and Unix) C. The first two-thirds covers the C language and C preprocessor; the last third focuses on C runtime libraries. The text has embedded small notes referencing the ANSI C standard where it applies.

One of the book's strengths is its coverage of pre-ANSI compilers and libraries. The authors note where ANSI and traditional C differ and provide solid suggestions for writing portable programs among various dialects. The runtime library references show prototypes for ANSI functions and comparable pre-ANSI C functions. I found the section on data representation, conversions, and byte ordering (big and little endian problems) very useful. This book is well organized by topic, which makes it an excellent reference.

Another fine older book from Prentice Hall is the second edition of ***The C Programming Language***, by Brian Kernighan and Dennis Ritchie. The first edition (published in 1978 and still available) achieved legendary status as the de facto definition of the C language. This 1988 revised edition was based on a draft of ANSI C, so it may be incorrect regarding the final ANSI standard in a few areas. Taking the tutorial approach, the text is illustrated with helpful examples using C's features. Although terse, this book is easy to read.

Another older title, Hayden's ***Portability and the C Language***, by Rex Jaeschke, is an excellent reference on advanced C issues and portability. This book focuses on porting C code between environments and provides useful problem-solving recommendations. The first part covers C language and preprocess portability; the latter half discusses runtime library routines. This is the only book I've found that covers complex and difficult C constructs well.

In the application area, I expect that Prentice Hall's ***Compiler Design in C***, by Allen Holub, will replace the Aho "dragon" book for would-be compiler writers. What makes this massive book stand out is its pragmatic nature. Profusely illustrated with real C code, it focuses primarily on lexical analysis, context-free grammars, and parsing. Holub briefly covers code generation and optimization. Complex concepts receive thoughtful and detailed examination. The text is interspersed with many working examples of a Lex work-alike and two parsers (a top-down LL and a bottom-up LALR), along with other useful C routines and programs. The example C code makes this book valuable even to those who are not interested in writing compilers. The source code is available on disk at additional cost.

BASIC AND NOT-SO-BASIC

Microsoft Press has a solid offering with the ***Microsoft QuickBASIC Bible***, by the Waite Group. This aptly named book is a complete and up-to-date reference on QuickBASIC (version 4.5) and also notes features or differences from many other BASIC dialects, including earlier versions of QuickBASIC. Its four sections cover core (basic constructs), multimedia (graphics, sound, and lightpen/joystick), devices (files, keyboard, printer, and serial ports), and development (DOS, memory, mixed language, and debugging). The inside covers offer an alphabetized list of keywords with page numbers. Appendices include a synopsis of articles and programs available from Microsoft via Compuserve and Genie, along with brief descriptions of third-party routines that are available as add-on QuickBASIC libraries.

I'm still waiting for the new Turbo Pascal 6 titles to arrive. In the interim, I can recommend W.W. Norton's new ***Oh My! Modula-2!***, by Doug Cooper. This introductory textbook solidly covers the major features of Modula-2 via enlightening examples. The chapters on library modules, interfaces, packages, and data abstractions (the essential strengths Modula-2 added over Pascal) are especially helpful. The book is available bundled with a disk of example programs (with test data) and a version of the Stony Brook QuickMod Modula-2 programming environment.

A fairly old Fortran book that I always like to recommend is Academic Press's ***Fortran Optimization***, by Michael Metcalf. This fine book shows Fortran programmers how to write optimized code from the start instead of depending on the compiler. Metcalf covers various aspects of optimization, several mainframe Fortran compilers (IBM and CDC), and Fortran portability. I'd love to find a comparable book on C optimization.

Wiley's ***Fortran Tools for VAX/VMS and MS-DOS***, by Jones and Crabtree, focuses on how to write software tools. The book illustrates modern FORTRAN 77 programming by developing a series of software tools that are based on the classic ***Software Tools***, by Kernighan

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

AMX

REAL-TIME MULTITASKING KERNEL

8086/88, 80x86/88 80386
Z80, 64180, 8080/85 68000/10/20

- Fast, reliable operation
- Compact and ROMable
- PC peripheral support
- DOS file access
- C language support
- Preemptive scheduler
- Time slicing available
- Configuration Builder
- Complete documentation
- Intertask messages
- Message exchanges
- Dynamic operations
  - task create/delete
  - task priorities
  - memory allocation
- Event Manager
- Semaphore Manager
- List Manager
- InSight™ Debugging Tool

THE BEST

Join over 600 developers such as IBM®, Xerox, Hewlett Packard, Hayes, Hughes Aircraft and NASA.

CHOOSE AMX

The best low-cost, high-performance real-time multitasking system available today.

No Royalties
Source Code Included

Demo Disk	$25 US	*Call for prices for other processors.*
Manual only	$75 US	
AMX 86	$3000 US	

(Shipping/handling extra)

IBM is a registered trademark of IBM Corp.
Z80 is a trademark of Zilog, Inc.
AMX, AMX 86, InSight are trademarks of KADAK Products Ltd.

KADAK Products Ltd.
206-1847 West Broadway
Vancouver, B.C., Canada
V6J 1Y5
Telephone: (604) 734-2796
Fax: (604) 734-8114

Circle 40 on Reader Service Card

and Ritchie (Addison-Wesley, 1976 and 1981). The book is informal and pragmatic. The authors are generous with their use of working example code, including string-handling routines, hashing functions, a Fortran preprocessor, a VAX Fortran-to-FORTRAN 77 translator, a lexical analyzer, a simple compiler, and a text file I/O system with redirection based loosely on C. Source code primitives that support both VAX/VMS Fortran and Lahey's F77L DOS compilers are provided in the appendix. The source code can be bought from the authors.

Another Wiley title, ***Programmer's Guide to Fortran 90***, by Brainerd, Goldberg, and Adams, was written by members of the ANSI and ISO committees. Note that this book is off the press before the ANSI standard has been approved. It focuses mainly on writing programs that use Fortran 90's new features. The book is designed more as a gentle textbook than as a reference work, with short illustrative examples.

FOR THE HOTTEST GRAPHICS

Addison-Wesley has two fine new second-edition books. The ***Programmer's Guide to the EGA and VGA Cards***, by Richard Ferraro, is an excellent text on low-level EGA and VGA programming. Almost twice the size of the original edition, this book covers SuperVGAs in detail. The first dozen chapters are updated from the original; the last nine are new and cover various SuperVGA chipsets. Programming examples are mostly in assembly language, with interfaces to C.

Ferraro's discussion of EGA and VGA programming is great, but the real strength of this edition is the coverage it gives to SuperVGA. Two long chapters are devoted to general SuperVGA hardware, software, and code basics—memory-addressing techniques, VESA, SuperVGA standards, etc. The remaining seven chapters cover chipsets from ATI, Chips and Technologies, Genoa, Paradise, Trident, Tseng Labs, and Headland (Video 7) in detail. Each one offers exceptionally detailed hardware-level discussions with example code. The appendices are also excellent, providing extended register mapping and separate tables for alphanumeric and graphics modes organized by mode number, manufacturer, and number of colors. The example code is available on disk. If I could own only one book on EGA/VGA/SuperVGA programming, this would be my book of choice.

The other second edition from Addison-Wesley is ***Computer Graphics***, by Foley, van Dam, Feiner, and Hughes. This is your state-of-the-art reference on 2-D and 3-D graphics. Chapters cover algorithms for basic and advanced raster graphics, curves and surfaces, solid modeling, achromatic and colored light, and illumination and shading. If you need to find the latest and fastest algorithm for, say, clipping 2-D lines (the Nicholl-Lee-Nicholl algorithm published in 1987), this is the place to look. Other chapters focus on typical graphics hardware, geometric transformations, graphic objects handling, input devices, user interface software, dialog design, image manipulation, and animation. Illustrative code fragments are in Pascal or pseudo-code. This reference is indispensable for writing efficient low-level graphics code.

PANDORA'S DOS

For DOS aficionados, Addison-Wesley has a winner in ***Undocumented DOS***, by Schulman, Michels, Kyle, Paterson, Maxey, and Brown. This book attempts to document all of the undocumented ("reserved," in Microsoft jargon) MS-DOS functions and quirks. The core chapters uncover DOS resource management (memory, processes, and devices), the DOS file system and network redirector, memory-resident software, command interpreters, and the DOS debugger interface. I found the chapter on the DOS file system and the network redirector to be the most fascinating. Versions of DOS through 4.0 are covered. Although much of this material has been published previously in magazine articles, some was new to me.

The appendix provides a listing of undocumented calls with explanations of their use and calling conventions, if known. Two disks are included; they contain a hypertext database of all the MS-DOS functions and sample programs illustrating the use of undocumented functions. An interrupt "spy" program is also included to determine which (undocumented) interrupt calls a program uses.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Effective marketing presentations

By William H. Roetzheim

As a consultant, you will probably have many opportunities to conduct marketing presentations. Presentations can be used to sell a proposal to a client, to inform potential clients at a professional seminar or conference, or to demonstrate a software or design concept to a current customer. No matter what the context, how well you do your presentation will have a large impact on how much money you pocket at the end of each billing period.

Any presentation can be enhanced by visual aids. These might include slides, overhead transparencies, or flip charts. I prefer overhead transparencies because they can be carried easily on a plane and because they lend themselves well to promoting group discussion as each one is displayed. Slide shows might have a more professional appearance, but they don't stimulate informal discussions.

PREPARING VISUAL AIDS

No matter what sort of visual aid you choose to work with, your preparation is the same. First you sketch out your figures on 8½-by-11-inch sheets and prepare your discussion around these pages. Rough sketches can be turned into finished 8½-by-11 pages using either a large font on your laser printer or a drawing/presentation package. I like a product called Arts and Letters Editor (available from Computer Support Corporation, 15926 Midway Road, Dallas, TX 75244; (214) 661-8960) because its large library of predrawn icons and pictures makes it simple to add visual interest to slides. Printed output can be photocopied onto blank transparency

ILLUSTRATION BY JIM CARPENTER

sheets to produce overhead transparencies that are ready for projection. There are also companies that will take output files from presentation software (including Arts and Letters Editor) and produce full-color transparencies for about $7 per overhead.

When preparing any visual aid, the most important point to remember is that each picture must make one, and only one, point. In reports, illustrations can safely contain two or even three interesting facts, because customers can spend more time examining a written illustration than they can a visual aid. Illustrations quickly lose their impact when they are projected onto a screen, so trying to cram information onto the screen is false economy.

Each transparency should tell a story without requiring any support from you. Your talk should expand on areas related to the illustration; leave the details for your verbal discussion.

The best method of preparing effective slides is to make a list of what point or story each slide will convey. You can then use this list to prepare or select the appropriate illustrations. Don't make the all-too-common mistake of sketching out or selecting illustrations first and then trying to decide what point they are making.

If you are used to speaking before a group, I probably don't have anything new to tell you. If you haven't had many opportunities to speak before groups, some hints might be in order.

PUBLIC SPEAKING TIPS

Start and finish your discussion on time. Make the content of your discussion serve your customers' needs, not your own. Every word you say should be for their benefit, not yours.

When preparing your briefing and also when presenting it, think in broad concepts, not specific details. By the time you give the presentation, you will be talking about something you know like the back of your hand. The details will come automatically. While speaking, be enthusiastic and animated. Move around the front of the room. Maintain eye contact with your audience—enthusiasm is catching.

If you feel a bit nervous, have someone else prepare a short (5–10 minute) discussion of some narrow technical issue. After you have talked for a short time, introduce the other speaker's topic and say, "Ms. Jones can explain this concept much more eloquently than I can." This will give you a breather to clear your throat, gather your wits, and dry your sweaty palms. Always have a cup of coffee or a glass of water by your side. If someone asks you a tough question, take a sip while you collect your thoughts, then answer.

Use terms that all of your audience will understand. Even more important, use terms that you understand. Don't use technical jargon just because your company's technical gurus use it. Be ready to admit that your technical people "handle those matters"; just make sure that one of them is in the room to answer questions.

It's good to include an example to support each point covered. Note that I said example, not proof. Customers really like to hear statements such as, "This specific problem can be solved using such-and-such a technique, as we discovered when implementing a similar system for so-and-so."

William H. Roetzheim is a senior associate with Booz, Allen & Hamilton, and a noted author and speaker on software engineering topics. This article is based, with permission, on the author's book, Proposal Writing for the Data Processing Consultant, *Prentice-Hall, 1986. Bill can be contacted at Booz, Allen & Hamilton, Inc., Suite 109, 4025 Hancock Street, San Diego, CA 92110; (619) 223-2307.*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

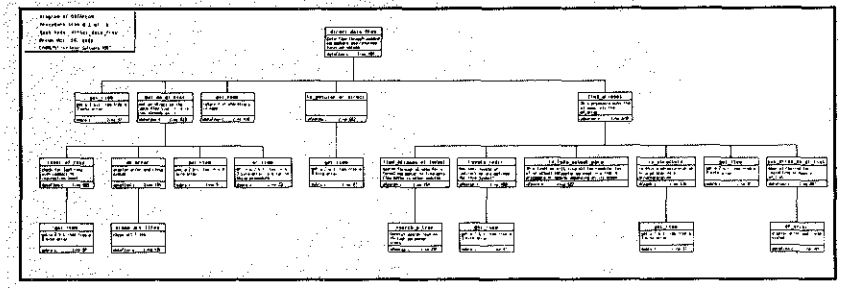
It's no sin not to know the answer to a question. Simply tell the questioners that you will get back to them later that day, or have someone else in the room answer. Most groups have technical people who are not happy until they can prove they know more than you do. Technical people get their satisfaction either by catching you in an outright error or by getting you to admit that they thought of a question that you could not answer.

If someone is annoying and persistent, tell them you will speak with them during a break. If someone is antagonistic, you might be able to take the heat off of yourself by turning the question or comment over to the audience (e.g., "Mr. Smith thinks that . . . . What do all of you think?"). Usually, someone in the audience will defend your position, keeping you out of the line of fire. Don't be afraid of allowing a minute or two of silence while waiting for a defender to appear. (Longer than that, and maybe you should rethink your position!) You can then step in as the hero and "rescue" the meeting by breaking up the discussion and continuing your presentation.

Have handouts that summarize your briefing, and distribute them at the end. And finally, always close any presentation with impact. The last sentence you say is probably the most important, so make it memorable.

C/ANALYST...

...THE HIGH POWERED DOCUMENTATION TOOL



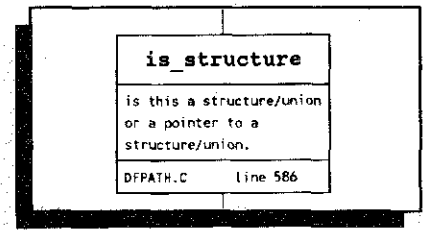
You have been handed a few inches of source code and have been asked to modify it. C/ANALYST can quickly bring you up to speed. It analyzes your C source and forms a database representing your entire program. From this database, C/ANALYST exhibits your program from different perspectives.

C/ANALYST is a set of four programs: ANALYZE, XREF, PROCTREE & DATAFLOW.

ANALYZE forms the database from your C source files. It has special features to process large multi-file programs quickly. Its built-in preprocessor saves include files in virtual memory as a stream of tokens to eliminate rescanning and redundant accesses to disk. Large hash tables provide quick access when there are thousands of symbols present in the code. Using virtual memory, ANALYZE can build databases much larger than the available memory.

XREF generates a function index followed by a global cross reference listing. The function index alphabetically lists each function, gives its location and its definition taken from comments in the source code. The global cross reference not only shows where each object is used but how it is used as well. Read and Write codes along with indirection levels are included with each reference. Objects are referenced to the function which uses them. Structure members are listed individually.

PROCTREE generates procedure tree diagrams which show the overall control flow of the program. Diagrams can be printed on HP LaserJet printers or sideways on dot matrix printers. Although the diagrams can be almost unlimited in size, a heuristic pruning method chops trees into multiple diagrams of reasonable size.



DATAFLOW searches through the database to find the flow of information between each module pair in the program. It shows flow through global variables when one module writes to a variable and another module reads that variable. Parameter passage to functions can be shown along with symbolic function returns. A sophisticated technique searches for indirect data passage through formal parameters which are pointers.

FOR MORE INFORMATION CALL US TODAY

(503) 581-5622

VERSION 2.0 $250.00 ppd.
+ 15.00 for overseas

30 day money back guarantee
requires MS-DOS 2.0 or above.

CATER SOFT WARE

2182 Westfarthing Way NW
Salem, OR 97304

Contact this advertiser directly.

FORTRAN PROGRAMMING TOOLS

FORWARN -- lint for Fortran, plus calling trees and cross references. **DEVT TOOLS** -- pretty printer, GOTO restructurer, etc.

For DOS, $329. and $129.

Also for OS/2 and Unix.

Quibus Enterprises, Inc.
3340 Marble Terrace
Colorado Springs, CO 80906

(719) 527-1384

Circle 47 on Reader Service Card

Developer's Office Management Software

Turn Your Office Productivity Around!

Easy to use computerized system to organize, operate and control your software business. Written for DOS and DESQview API. ***The Developer's Assistant*** also works with Windows and Software Carousel.

Major features:	Other features	Affordable
User and Dealer indexed database		
"Bug" look-up DB for software support		
Superb invoicing capabilities		
Utilities i.e., serial # embedding program		
Directory of packagers, suppliers, etc.		
Directory for product releases		
Lots of reports and other features		
Supports both laser and dotmatrix printers		
Free 900 ☎ number to support your software		
Built-in calculator, phone dialer, other accessories		

Powerful • Integrated • Versatile • Slick

(512) 834-8927 $129.00 (+ tax in Texas)

(VISA/MasterCard/AMEX accepted)

S.L.A.M. Software
3202 W. Anderson Ln. Ste. 208-176
Austin, TX 78757-1022

Circle 48 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

COBOL revisited

By Kipp Martell
and Tim McCormley

Not long ago, while I was riding a shuttle bus at a recent trade show, I overheard two gentlemen in the seat in front of me talking about "an archaic computer language." I'm not usually one to eavesdrop, but the word "COBOL" attracted my interest. Evidently, they had passed by a vendor who was selling a compiler and were amused by his product. "COBOL! Who programs in COBOL these days?" The answer I offered them is, "Lots of people!"

There are several hundred million lines of existing COBOL code. COBOL is used by thousands of programmers throughout the world. Ford Motor Company has a staff of 1,500 COBOL programmers, Bank of America has 3,000, and the federal government employs over 60,000. That's in the U.S. alone. COBOL is an international language of management information systems and data processing, and it has been estimated that 45% of all programmers in the world program in COBOL.

COBOL was originally designed to develop large financial applications requiring the storage of millions of records. Therefore, it has built-in structures to handle data manipulation. Over the years, as more interactive processing has become popular, COBOL compilers have been designed with the aim of providing tools to create "user friendly" interactive applications.

COBOL'S STRENGTHS

COBOL has some features that do not exist in other popular languages such as C, Pascal, and Fortran. These include a built-in file system that supports sequential, relative (random access), and indexed files; true decimal math (absolute accuracy); a built-in SORT verb; and built-in screen printing extensions. In addition, complex conditions can be represented to specify discontinuous ranges (COBOL level 88 codes).

COBOL comes with its own record-oriented file system. In Fortran, Pascal, and C, you would have to create your own file system. Needless to say, creating a database or record management file system can take a great deal of time.

A major feature in COBOL is that math functions are handled as true decimal (base 10) math. This allows $10.33, for example, to be represented exactly. In other languages (C, Pascal, and Fortran), the real data type uses binary (base 2) math and can't represent most decimal numbers with complete accuracy. This has obvious consequences for writing financial applications.

Additionally, COBOL implementations on microcomputers have a very intuitive method for displaying on-screen information. Consider the following code fragment:

```
DISPLAY CUSTOMER-NAME LINE 10 COLUMN 1
        ERASE END SCREEN.
```

It is fairly easy to tell what this code will do when it is executed. Consider the same functions in C, Pascal, and Fortran:

```
C:        printf("%s",customer_name);
PASCAL:   writeln(customer-name)
FORTRAN:  write(100,customer-name)
          format (10x, 30A)
```

The only problem is that these languages do not support a standardized way to reference the screen; the customer name simply appears on the screen at the current cursor position.

Additionally, COBOL is easy to learn because it uses English-type words instead of esoteric symbols. As an example, try to imagine what the following C code fragment might do:

```
for (i=0; i<MAX; i++)
   printf("%4d%c", a[i],(i+1) % 10 ? ' ':'\n');
```

This code prints a table to the terminal screen. Contrast this with an equivalent COBOL statement:

```
perform varying idx from 1 by 1 until idx > MAX
    display a-table(idx) line 3 column idx
end-perform.
```

COBOL is weak in that a pointer type is not part of the language standard, so dynamic data structures (trees, linked lists, etc.) are difficult to represent.

NEW ENVIRONMENTS

Traditionally, COBOL was used on the IBM mainframes, but it has been used recently in other environments, including Domain and HP/UX. Several of the most popular DOS-based accounting packages are written in COBOL, such as "Real World," MCBA, and Macola. In most cases, these products were ported from minicomputer packages.

Six years ago, ANSI published the COBOL-85 standard. Several added features encourage structured programming techniques, as well as increasing the performance of applications written in COBOL. Many language-development tools have been designed that make COBOL program development easier, including code generators, screen generators, windowing capabilities, etc.

Most of the COBOL compilers on the market feature several important extensions to the '85 standard. Aside from fast compilations, some offer an interface to relational database file systems, embedded SQL, built-in source-level debugging, pop-up windowing, and color and line drawings that greatly enhance the user interface.

A programming language is just a tool. A tool is used for a specific application; you wouldn't use a sledgehammer to kill a flea. I may use a language other than COBOL to write an operating system; however, I would not write a large financial application in C or Fortran. Using the right tool for the job is the key. A hammer doesn't tighten nuts and bolts very well, and a wrench isn't intended to drive nails—but using them for what they were designed for can produce great results. The same is true of COBOL, C, Pascal, and Fortran.

I think those guys on the bus might even agree with me.

Kipp Martell is marketing relations manager and Tim McCormley is technical support manager at Acucobol. They can be reached at 7950 Silverton Avenue, Suite 201, San Diego, CA 92126; (619) 689-7220.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Language parsers and Fortran tools

By M. Steven Baker

Most linguists believe that language learning is innate to the human species. I wish fluency with a new computer dialect could be achieved as easily as saying "mama"; unfortunately, it's not. These short reviews focus on software products designed to work with computer languages. I'll touch first on language generators. I'll also examine some Fortran tools since, with the imminent adoption of Fortran 90, this language will see some major changes.

PARSER GENERATORS

Scanner and parser generators (often called compiler-compilers) are designed to ease the development of programs that process languages and input. Made famous by Unix's **lex** and **yacc**, these products are usually associated with writing the front ends of compilers. However, both scanning input for tokens and parsing languages for proper grammar can be used for a wide range of operations, from "Implementing Dynamic Data Exchange in Windows" (see Bill Hall's article in *PJ* 8.1) to writing command shells.

These sophisticated tools are not for the weak of heart: the learning curve can be steep. Several hurdles await the uninitiated—learning the program syntax, writing and debugging your grammar. One has to become fluent with the concept of grammars and how they are expressed, usually in Backus-Naur Form (BNF). But once you've made it past these hoops, the rewards can be almost limitless.

Along with a family of other Unix-like tools, **Mortice Kern Systems** (MKS) sells *MKS LEX* and *YACC* for DOS and OS/2. These compiler construction tools emulate their Unix counterparts. LEX is a scanner generator that takes a stream of characters as input and then outputs a stream of tokens, such as identifiers, operators, and keywords. Like the other products reviewed, MKS LEX can handle the full 8-bit character set. LEX is generally used to segment the input in preparation for parsing with YACC. MKS YACC has full support for LALR grammars and can be used to create multiple parsers in a single program, such as SQL and dBASE syntax.

The DOS version ($249) is a small-model program and runs out of memory (dies) on large grammars (Ada, COBOL, C++, dBASE4, etc.). The OS/2 version ($349) doesn't have these problems. MKS programs generate the smallest parse tables at reasonable speed. Of the three products I worked with, MKS has by far the best manual, available separately ($50); it's worth having even if you choose a competing product. Alas, MKS comes with only a few sample grammars. They're good for learning from but won't help you finish an SQL parser in a weekend.

Abraxas Software sells *PCYACC* for DOS, OS/2, and Mac. PCYACC includes LEX and YACC programs that emulate the Unix utilities along with ANSI C and other example grammar files. File examples also show how to incorporate error recovery. The Professional package adds a number of sample grammar files including C, C++, COBOL, dBASE, Fortran, HyperTalk, Pascal, Prolog, PostScript, SQL, and Smalltalk.

Under DOS, PCYACC is a large-model program and can handle most grammars. PCYACC runs the slowest when generating parser tables and creates larger tables. The Professional edition adds PCY286, a version using the Ergo 286 DOS extender to execute faster and swallow larger grammar files. This version includes many grammar files, but some are not complete or debugged. For example, the dBASE and Fortran grammars I tried would not swallow modest fragments of real (bug-free) code. The Personal version is $249, the Professional version is $495, and the OS/2 version is $695.

LALR Research sells *LALR*, a language-development toolkit for DOS and OS/2. The Personal version includes DFA (a lexical generator), LALR (a parser generator), YACCTRAN (a YACC to LALR translator), VIEW (a table viewing utility), and some small sample grammars. The Professional version adds grammars, including C, C++, BASIC, COBOL, dBASE, Ada, Modula-2, Pascal, Fortran, PL/1, and SQL.

Although LALR is designed to accomplish the same goals as Unix's Lex/Yacc combination, the toolkit uses a different input language. YACCTRAN will convert most YACC constructs into an LALR grammar. The manual covers program usage and the DFA and LALR grammar input format. LALR-generated parsers have built-in error recovery. Multiple parsers in a single program are supported. LALR generates parser tables blazingly fast. The parser tables are a bit smaller than PCYACC's. The VIEW utility is without equal for debugging a grammar. It allows you to interactively view a grammar table on-screen and pick legal constructs from menus using arrow keys. LALR Professional v4.1 is

VENDOR INFORMATION

LALR Professional
LALR Development
Burnt Mill Road
Tustin, CA 92680
(714) 832-2274
Circle 247 on Reader Service Card

MKS LEX and YACC
Mortice Kerns Systems, Inc.
35 King Street North
Waterloo, Ontario, Canada N2J 2W9
(519) 884-2251
Circle 248 on Reader Service Card

PCYACC Professional
Abraxas Software, Inc.
7033 SW Macadam Avenue
Portland, OR 97219
(503) 244-5253
Circle 249 on Reader Service Card

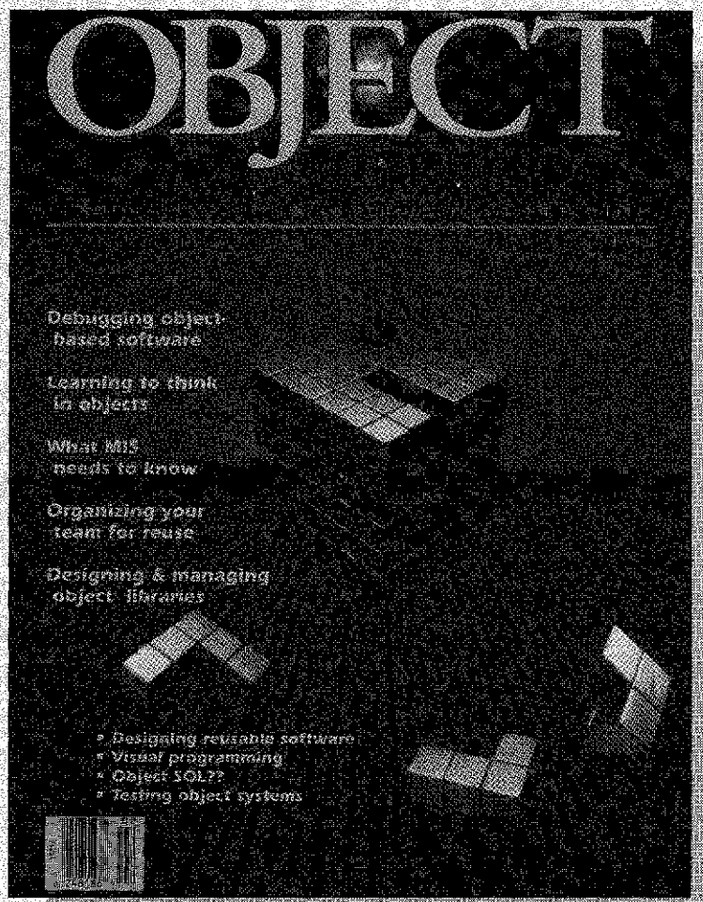
FOR_STRUCT
Cobalt Blue
2940 Union Avenue, Suite C
San Jose, CA 95124
(408) 723-0474
Circle 250 on Reader Service Card

Forwarn & Fortran Development Tools
Quibus Enterprises, Inc.
3340 Marble Terrace
Colorado Springs, CO 80906
(719) 527-1384
Circle 251 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EVERY PIECE OF INFORMATION IN OUR MAGAZINE IS REUSABLE.

Premieres May 1991



The concept of reuse is quickly expanding on an international level. Reuse provides for a more efficient, less expensive environment. Object-based development will improve the quality of your work by avoiding redundancy of efforts and duplication of costs—saving you valuable time and money.

Object Magazine is an international publication targeting the field of object orientation. Every issue is dedicated to informing both experienced and beginner computer professionals on getting the job done quickly and effectively using object technology. We discuss analysis and design methodologies, language comparisons, developmental environments, database advances and project management techniques. All are based on real-world applications, extractable ideas and advice that you can use again and again in your object-based software implementations.

Your environment is complex. ***Object Magazine*** strives to simplify it by supporting the widespread growth of object development and reuse.

Object Magazine
Bringing you the next generation in software development.

SPECIAL CHARTER SUBSCRIBER OFFER Circle 49 on Reader Service Card

❑ Yes, sign me up as a charter subscriber to ***Object Magazine*** at the special low rate of $25, a savings of 14% off the regular rate.
❑ 1 year (6 issues) - $25 ❑ 2 years (12 issues) - $40
(outside the U.S., add $24 a year for air service)

OBJECT magazine

ethod of Payment (foreign orders must be prepaid in US dollars drawn on a US bank)
❑ Check enclosed (payable to ***Object Magazine***)
❑ Bill me ❑ Charge my: ❑ Visa ❑ Master Card
ard # _____ Exp. date: _____
ignature: _____

Name: _____
Company: _____
Address: _____
City/State/Zip: _____
Telephone: _____

Object Magazine, Subscriber Svcs., Dept. OBJ, PO Box 3000, Denville, NJ 07834; fax (212)274-0646. Offer expires 12/31/91. **D1CPJ**

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

$495 for DOS, $695 for OS/2; the Personal version is $249.

SOME FORTRAN TOOLS

Although FORTRAN 77 has been the standard for more than a decade, many large existing FORTRAN programs were written using FORTRAN 66 syntax. Since the 1970s, DEC (VAX FORTRAN) and other vendors have incorporated new language features into their compilers. Many of these will become part of the Fortran 90 standard.

Cobalt Blue has released *FOR_STRUCT v1.2*, a Fortran restructing utility to transform spaghetti FORTRAN into fully structured code. FOR_STRUCT converts older IF-GOTO, GOTO, and DO statements to equivalent IF-THEN-ELSE, DO-WHILE, and DO-ENDO constructs. This version can also generate Fortran 90's SELECT CASE, EXIT, and CYCLE syntax.

The product consists of FSTRUCT (the restructuring tool) and FTRANS (a Fortran preprocessor). Various Fortran dialects are supported. FSTRUCT can restructure code to FORTRAN 77, VAX Fortran, MIL-STD Fortran, or Fortran 90. It can convert holleriths to quoted strings, indent code for structure, and can upper- or lowercase keywords, comments, and variables. The PC version has some routine size limits (about 1600 lines maximum per module) that don't exist in the Unix versions. This is an excellent product for improving older FORTRAN code and costs $825 for DOS, $1250 for Unix/386, and $1350 for SUNs.

Quibus Enterprises has two products for Fortran programmers. *Forwarn*, a static analyzer, detects many common programming errors, including incorrect parameter lists, mismatched common blocks, misspelled variable names, and loss of precision. Forwarn can also generate detailed cross-reference listings and call-tree diagrams. It can analyze a single file or a directory of source modules. Three levels of analysis messages—information, warnings, and errors—can be individually enabled both locally (a single procedure) and globally (across routines and files). Forwarn had no problem handling very large files. This program is indispensable for cleaning up older Fortran code.

Fortran Development Tools are utility programs that aid in handling Fortran source files. They include PRETTY (a formatter), OUTLINE (a listing generator that highlights loops), PREP (a Fortran preprocessor), SPLIT (splits routines into files), COMBINE (merges files), SEQ and UNSEQ (adds or strips sequence numbers). The PRETTY formatter provides some restructuring. The DOS version has size limits of routines that can be restructured. The other programs are simple but useful utilities.

Forwarn is $329 for DOS or Unix/386, $349 for OS/2, and $1200 for workstations. The Fortran Development Tools are $129 for DOS or Unix/386, $149 for OS/2, and $500 for SUNs.

OPUS
SOFTWARE
"Without a doubt, OPUS Make is the hottest make utility on the market"
Tom Swan, PCWorld
For professional programmers, OPUS Make is the superior choice in a make utility. It is the fastest most full-featured make utility there is. Features include:
■ DOS version uses only 3K memory!
■ Multiple directory support.
■ Supports: Polytron PVCS™
Burton Systems TLIB™
Microsoft LIB™
SLR Systems OPTLIB™
■ Generates automatic response files of unlimited length for LINK and LIB.
■ OPUS MKMF included: An automatic dependency generator that fully understands C preprocessor directives.
Both DOS and OS/2 executables for only: $129 MC, Visa, COD, or PO. 60 day MBG!
Call 1-800-248-OPUS
International: 415-864-7981
1032 Irving Street, Suite 439, San Francisco, CA 94122
Trademarks owned by their respective companies
OPUS MAKE

Circle 50 on Reader Service Card

Conversion Tools for Fortran Programmers
■ FOR_STRUCT™
Transforms spaghetti FORTRAN into fully structured code, with or without VAX and FORTRAN-90 extensions! GOTO and IF-GOTOs are replaced with IF-THEN-ELSE, DO-WHILE and DO-ENDDOs, while retaining programming logic. Code is not duplicated, dead segments can be removed and different style options are offered. Fully compatible with FOR_C.
■ FOR_C™
Converts standard FORTRAN and many VAX extensions (e.g. structures) into ANSI C. Produces extremely readable and maintainable C code, with excellent I/O and character translations. FOR_C also utilizes a C-style preprocessor and C prototypes, and includes source to extensive runtime libraries. New version 3.1 now shipping.
■ Call for more information and combined product discounts!
COBALT BLUE
2940 Union Avenue, Suite C
San Jose, CA 95124, USA
TEL(408) 723-0474,
FAX(408) 377-7648

Circle 51 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ PRESSROOM

CC-RIDER/C++ RIDER

Western Wares has come out with C++ RIDER for source code analysis and online help with C++. The C++ RIDER package consists of a pop-up source code browsing utility (CCRIDER) and a source code analyzer (CCSYM). CC-RIDER can be used with any text-mode editor, integrated environment, or debugger to provide hypertext symbol referencing while programming. A hot-key brings up the source code and makes it possible to edit or paste any symbol definition. CCSYM performs analysis and symbol database management, with the ability to create a Microsoft QuickHelp database and generate function prototype header files for C++. Detailed symbol reports can also be generated. C++ RIDER runs under DOS and OS/2. It is compatible with ANSI C and with the proposed ANSI C++ standard. The Standard Edition is $89 and the Professional Edition is $279.

Western Wares
Box C
Norwood, CO 81423
(303) 327-4898
Circle **227** on Reader Service Card

INTEK C++ FOR WINDOWS

Intek Integration Technologies is shipping Intek C++ 2.0. Intek C++ is a translator that generates C code output from C++ source. The DOS version runs in protected mode and supports all memory models for DOS. A range of C compilers are supported, including those from Microsoft, Borland, Watcom, and MetaWare. Intek C++ fully supports the extended keywords (**far**, **huge**, **pascal**, etc.) along with those added with Microsoft C 6.0. It runs under MS-DOS on 80386 and under Unix System V/386. Intek C++ sells for $495.

Intek Integration Technologies
1400 112th Avenue SE, Suite 202
Bellevue, WA 98004
(206) 455-9935
Circle **228** on Reader Service Card

32-BIT C++ COMPILER

Zortech is releasing Zortech C++ DOS 386, a 32-bit C++ compiler that generates 80386 object code compatible with the development tools from Phar Lap Software. The Zortech C++ Workbench comes with the package. This is a multi-windowed editing environment with mouse support. A C++ debugger features mouse support, breakpoints, watchpoints, and memory and heap protection. Zortech's C++ Tools Library with 32-bit support is also provided. Other features are the Flash Graphics library, runtime library support for the Microsoft mouse, 80387 math coprocessor support, a floating point emulation library, a make facility, and an object code disassembler.

The Developer's Edition is available for $995. Source code for the runtime libraries is provided at no extra charge.

Zortech, Inc.
4-C Gill Street
Woburn, MA 01801
(800) 848-8408
Circle **229** on Reader Service Card

PC-METRIC FOR MODULA-2

SET Laboratories is now shipping version 3.0 of PC-METRIC for Modula-2. PC-METRIC computes how difficult a computer program will be for a programmer to understand, test, or modify. Supporting software science and span-of-data reference measures, Version 3.0 also counts blank lines, comments, average variable name length, and comment density.

The CONVERT tool imports PC-METRIC reports into Lotus 1-2-3, and a companion set of macros facilitates analysis of the metric data. The PIE-CHART tool allows users to graphically present their data without access to Lotus 1-2-3. PC-METRIC is also fully compatible with DMG's Software Metric Repository, allowing analysis of software metric information across projects.

Version 3.0 of PC-METRIC for Modula-2 is $199. Other language versions are also available.

SET Laboratories
P.O. Box 868
Mulino, OR 97042
(503) 829-7123
Circle **230** on Reader Service Card

PROWINDOWS FOR QUICKBASIC

DSE Software Publishing announces ProWindows Professional 2.0, a software development system and add-on library for Microsoft QuickBASIC and BASIC PDS 7 compilers. The new version has been redesigned to support an event-driven architecture, minimizing the need to manage complex user interfaces. Programmers can write code to process a particular event, leaving much of the user-interface mechanics to ProWindows.

This updated version includes an advanced window manager supporting multiple text-based windows. The Data Entry Manager facilitates creating dialog boxes and complex data entry screens, and it controls data validation.

ProWindows Professional is $59.95, including a QuickBASIC or BASIC PDS 7-compatible library, the ProDesigner Integrated Development Environment, and a disk-based user's guide, tutorial, and quick reference.

DSE Software Publishing
P.O. Box 96
Willits, CA 95490-0096
(707) 459-4358
Circle **231** on Reader Service Card

32-BIT FORTRAN

Lahey Computer Systems is releasing Version 4.0 of F77L-EM/32. This 32-bit FORTRAN comes bundled with the OS/386 Ergo DOS extender. F77L-EM/32 also works with Ergo's OS/386 VM (virtual memory) kernel, making it easier to write and port large programs. New features include support for some Fortran 90 constructs, an improved editor, a **make** utility, and new manuals.

Runtime versions of your application can be distributed for free. The virtual memory kernel is bound into the application at no additional cost. F77L-EM/32 is $895, and Ergo OS/386 $395.

Lahey is also upgrading Lahey Personal FORTRAN to Version 3.0. New features include F77L-EM/32's editor, a **NAMELIST** and **nargs** function, VAX Fortran control statements, conditional compilation, and DESQview compatibility. Lahey Personal is $99, the Lahey Personal Toolkit is $59, and the two bundled programs are $129.

Lahey Computer Systems
865 Tahoe Blvd.
P.O. Box 6091
Incline Village, NV 89450-6091
(800) 548-4778
Circle **232** on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LANGUAGES CONFERENCE

Borland International will host the first International Languages Conference April 28 through May 1, 1991, in San Francisco. Aimed at software developers and technical managers, the conference will feature presentations from industry leaders on topics such as developing applications for windowing environments, increasing programmer productivity, and object-oriented programming. The conference fee is $795, with group rates available.

CT Meeting Planners
Borland International
1800 Green Hills Road
Scotts Valley, CA 95066-0001
(800) 942-8872
Circle **233** on Reader Service Card

FAST SLR WINDOWS LINKER

SLR Systems has released OPTLINK/WINDOWS 2.5, a segmented linker designed to generate DOS, Windows, and OS/2 EXE and DLL files. OPTLINK/WINDOWS is one of the fastest linkers on the market, performing up to 20 times faster than Microsoft's LINK. In addition, OPTLINK/Windows sets no internal limits on the number of modules, libraries, publics, or segments being linked and uses EMS and XMS when available. The linker supports Codeview 3.0 and Windows 3.

SLR has also updated its OPTLINK/COMPRESS (DOS) linker to v2.50. The new version produces faster compressed self-extracting EXE files that automatically decompress when the file is executed. The decompression routine requires less than 1K of extra memory.

Both linkers will use EMS or XMS for virtual storage and disk files. OPTLINK/WINDOWS and OPTLINK/COMPRESS sell for $350 each.

SLR Systems
1622 North Main Street
Butler, PA 16001
(412) 282-0864
Circle **234** on Reader Service Card

MORE DOS MEMORY

V Communications has released MEMORY COMMANDER, a memory manager for DOS on 80386/486 machines. MEMORY COMMANDER can provide DOS applications with up to 900K of continuous memory. In order to free up space for programs with large data files, MEMORY COMMANDER moves TSRs and device drivers into high memory as the system is booted without requiring that the AUTOEXEC.BAT or CONFIG.SYS files be edited. MEMORY COMMANDER also maps the ROM BIOS into RAM for improved performance and provides full EMS and XMS emulation.

MEMORY COMMANDER is installed and configured via full-screen menus. Advanced users can select detailed options and different memory sizes and modify the memory maps. Once installed, MEMORY COMMANDER will automatically select the best of its five memory models for an application. MEMORY COMMANDER is available for $99.95.

V Communications
4320 Stevens Creek Blvd., Ste. 275
San Jose, CA 95129
(408) 296-4224
Circle **235** on Reader Service Card

ACCSYS FOR PARADOX

Copia International has expanded the capability of AccSys for Paradox to include C and C++ support for the Borland Turbo and Zortech compilers. Programmers can use this new version of AccSys to develop an application that runs under Windows 3.0 and OS/2 using the Microsoft C compiler.

AccSys gives the programmer control over Paradox table files and primary and secondary index files and expands the capability of the Paradox high-level language interface. Programs written with AccSys can be accessed from within Paradox or in stand-alone mode.

The single-user version is $395 ($795 with source code), and the multiuser version is $750 ($1500 with source). There are no runtime royalties.

Copia International
1964 Richton Drive
Wheaton, IL 60187
(708) 682-8898
Circle **236** on Reader Service Card

TOOLBOOK FOR OS/2

Asymetrix Corporation has released ToolBook 1.0 for OS/2. Initially released for the Windows graphical environment, ToolBook is a software construction set that allows users to build custom graphical applications without using traditional programming languages. ToolBook for OS/2 is file-compatible with ToolBook for Windows, so applications can be developed on one system and run in both environments.

ToolBook's application development tools include a color drawing package and an object-oriented programming language called OpenScript. These feature hypertext-style navigation capability, animation, text formatting, and flat-file database creation. OpenScript is extensible through Windows and OS/2 Dynamic Link Libraries.

ToolBook for OS/2 comes with a variety of applications, a hypermedia tutorial, a business calculator, an animation primer, and a collection of sample scripts, clip art, and page designs. ToolBook for OS/2 is $395, as is ToolBook for Windows.

Asymetrix
110 110th Avenue NE, Suite 717
Bellevue, WA 98004
(206) 462-0501
Circle **237** on Reader Service Card

WINDOWS TOOL FROM MATESYS

Matesys has new tools for Windows 3.0 developers. ObjectScript helps programmers develop applications for and port DOS applications to Windows. The program's objects and commands automate a number of Windows tasks such as designing screens, reports, and tables. ObjectScript can also be used to quickly prototype a new application. Versions are available for BASIC and C. A Database Applications add-on offers dBASE III file support.

ObjectView provides interface construction tools for creating custom SQL applications that run under Windows. Built upon a graphic editor, the tools include icons, pull-down menus, and tables. In addition, ObjectView provides a complete set of high-level functions for interaction with SQL engines for data validation and integrity. The applications created with ObjectView can be server-independent.

ObjectScript is $150 for the BASIC entry-level version and $495 for ObjectScript Pro (the BASIC or C versions).

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ObjectView BASIC is available for $899 and ObjectView C is $995.

Matesys Corporation
900 Larkspur Landing Circle, Ste 175
Larkspur, CA 94939
(415) 925-2900
Circle **238** on Reader Service Card

WINDOWSMAKER PLATFORM

Blue Sky Software is releasing WindowsMAKER Professional, a graphical platform for designing Windows applications. This "desktop programming" package allows users to prototype, animate, and test their Windows program in a WYSIWYG point-and-click environment. When the prototype is complete, WindowsMAKER produces code for standard programming languages and all the production files necessary to compile a native Windows application.

WindowsMAKER offers support for Microsoft C 5.1 and 6.0, Zortech C++ 2.1, and Glockenspiel C++ 2.0. There are no royalties or licensing fees. WindowsMAKER Professional is $995.

Blue Sky has also released Magic Fields 1.0, an object-oriented data validation extension to the Windows 3.0 environment. Magic Fields consists of a large collection of objects that perform data field validation. Magic Field objects can be used with any standard Windows dialog editor. There are no runtime royalties. Magic Fields is $295.

Blue Sky Software Corporation
2375 East Tropicana Ave., Suite 320
Las Vegas, NV 89119
(702) 456-6365
Circle **239** on Reader Service Card

SAGE CONTROL PAKS

Sage Software has released two application development libraries for the Microsoft Windows 3.0 and OS/2 Presentation Manager (PM) environments. Sage Control Pak/W and Sage Control Pak/PM provide developers with object-oriented, message-handling, reusable class libraries for building simple or complex user interfaces. Both packages also permit programmers to enhance the standard dialog controls offered by Windows.

The Sage Control Paks are best for developing applications in either C or C++ but can be used in compiled form with any Windows-compatible compiler. The Control Paks include both source and compiled code.

In compiled form, the Control Paks are dynamic link libraries (DLLs) and permit the sharing of dialog controls between applications. The source code allows a programmer to manipulate every aspect of each dialog control. The Control Paks cost $495 each or $595 for both.

Sage Software System
1700 NW 167th Place
Beaverton, OR 97006
(800) 547-4000
Circle **240** on Reader Service Card

"Writing a TSR is exceptionally easy" ... and now it's inexpensive too!

Now you can turn Turbo Pascal programs into rock solid TSRs with ease. **TSRs Made Easy** lets you create conventional TSRs or swapping TSRs that use only 6K of RAM. **TSRs Made Easy** provides ■ TSR swapping to EMS, XMS, or disk ■ selectable hot keys ■ keyboard macros ■ unloadable TSRs ■ 8087 TSR support ■ interface to transient programs ■ ISR handling, and more.

TSRs Made Easy includes full source, complete documentation, and plenty of small example and demo programs. You pay no royalties.

"Writing a TSR. . .is exceptionally easy. The documentation is extremely readable and well done."
Computer Language, May 1990

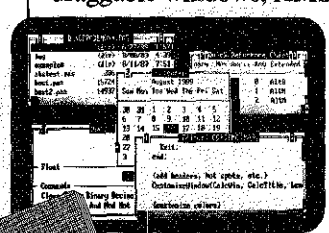
One of programming's most formidable tasks is now very simple. . . and very affordable!

TSRs Made Easy, only $49.

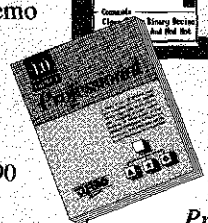
TSRs Made Easy has exactly the same TSR routines as OPro. TSRs Made Easy requires Turbo Pascal 6.0, 5.5 or 5.0. OPro requires Turbo 6.0 or 5.5. 5.25" and 3.5" disks included. Add $5 per order for standard shipping in U.S./Canada. Call for other shipping charges. Registered owners of OPro may update to version 1.1 for $20, include your serial number.

New! Object Professional for Turbo Pascal 6.0

Object Professional version 1.1 is fully updated for Turbo Pascal 6.0. New are SAA/CUA style dialog boxes, draggable windows, XMS/EMS 4 support, and more.



Object Professional includes over 100 object types that will multiply your productivity. Included are windowing and menu systems, menu and data entry screen generators, data object types, and routines that provide swapping TSRs.



You'll get up to speed fast with clear documentation, on-line help, full source code, and hot demo programs.

"The range of objects is fantastic. Object Professional could save you man-years of effort."
Jeff Duntemann

Object Professional, only $189.

Call toll-free to order: 1-800-333-4160

TURBO Power

9AM–5PM PST Monday through Friday, USA & Canada.
For more information call (408) 438-8608, fax to (408) 438-8610, or send mail to CompuServe ID 76004,2611
TurboPower Software PO Box 66747 Scotts Valley, CA 95067-0747

Circle **54** on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ BILLBOARD

FIGHT PIRACY & PROTECT YOUR PROGRAM $$$'s!

Since 1986, companies worldwide have been choosing Az-Tech security products. If you demand the strongest protection available, why not choose one of these "proven leaders":

- **EVERLOCK** Copy Protection
- **EVERTRAK** Software Security
- **EVERKEY** Hardware "Key" Software Security

For IBM and Compatibles.
30 day money back guarantee.
Free info and demo disk available.

Az-Tech Software, Inc.
305 East Franklin
Richmond, MO 64085
(800) 227-0644 (816) 776-2700
FAX (816) 776-8398

Circle **110** on Reader Service Card

QuickBASIC 4.5 - PDS 7
Software Development System

- Multiple Overlapping Windows with Full Mouse Support.
- Windows can be clicked, moved, resized, scrolled, etc.
- Popup & Pulldown Menus, Lists, Calculator, Calendar, Directories, and more.
- Dialog Boxes and Data Entry Screens.
- Context Sensitive Help Screens.
- Event-Driven Architecture.
- Supports MDA-VGA text modes.
- Integrated Development Environment.
- File, Directory, Keyboard, BIOS, and TSR functions.
- Disk-based Documentation.
- Only $59.95 (+CA Tax & $5.00 S/H)

DSE Software Publishing
Post Office Box 96 - Willits, CA 95490
707 459-4358

Contact this advertiser directly.

SUPER DiskCopy
Duplicates Disks Fast!

•Read diskette only once. •Quickly, accurately duplicates disks (360K, 720K, 1.2M, 1.44M) on your own PC/XT/AT/etc. © •Formats, copies, verifies, stores images, etc. •Even faster with pre-formatted disks. •Serialize disks and print labels, with included utility. •Replaces $1000s in hardware.

Only $79 +S&I. ($495• MULTI-USER)
S&I: $6 USA48; $10 N.Amer; $22 Foreign

SSW, 403 E. NASA, #377, Webster, TX 77598
Discover, VISA, MC, AX Accepted
713.488.0210 FAX:280.0525

Circle **111** on Reader Service Card

Write Multitasking DOS Programs with Conquerrent C Function Library

- Execute up to 16 C functions concurrently within one program.
- Great for real-time, data acquisition, clock displays, background printing, simulation, process control, games.
- Transparent reentrancy protection for most library functions.
- All memory models supported in Turbo and Microsoft C on DOS 80x86 PC's.
- 4K to 12K overhead.
- $99 complete, no royalties, 30 day satisfaction guarantee.

Available from:
Interstitial Software Systems
321 E. Buckingham Rd., Ste. 140
Garland, TX 75040

Circle **112** on Reader Service Card

PARADOX POPUP MENUS

Give your Paradox application an interface similar to PC Tools. Includes popup menus, dialog boxes, input/output boxes. Fast, tight, EASY. Includes PAL source code. Runtime and network ready. Only $45 plus $5 S/H.

PARADOX FIELD WRAPPING

Wrap text automatically down a Paradox field column. No more 256 character limit! This is a quick PAL procedure, and does NOT shell to DOS like competing products that require RUN BIG. Only $45 plus $5 S/H.

Hardware: 10 Mhz 8088 XT turbo/640k or better recommended.

Robin Rowe Software
1260 N. Dearborn, #310
Chicago, IL 60610-2240
312-280-7693

Paradox

Circle **117** on Reader Service Card

TLIB™ is FASTEST!

Times are to update a 45K library on a PC/XT. PVCS and TLIB 3.0 are from Sept 87 *PC Tech Journal*. MKS RCS 4.2 and TLIB 5.0 are newer.

TLIB™ is *BEST!*

"Do not be fooled by the fact that this is the least expensive of the five packages reviewed here - TLIB has features and power to spare" John Rex, **Computer Language**

"TLIB is a great system" J. Vallino, **PC Tech J**

- **Full-Featured Version Control for Software Professionals.** Check-in/out locking. Branching. Keywords. Wildcard and list-of-file support. Can *merge parallel changes and undo intermediate* revisions. Network and WORM support. **Mainframe compatible** deltas for Pansophic, ADR, IBM, etc.. Integrates with **Opus™ MAKE & Slick™ MAKE**

MS-DOS $139, OS/2 $195 + shipping Visa/MC
5 station LAN license $419 (OS/2 $595), call for other sizes

BURTON SYSTEMS SOFTWARE
PO Box 4156, Cary, NC 27519 **(919) 233-8128**

Circle **118** on Reader Service Card

Why you want BATCOM!

BATCOM is a batch file compiler that compiles your ".bat" files to ".exe" files to make them faster, more professional, and more capable. **BATCOM** extends DOS with new commands so you can read keyboard input, perform arithmetic, use subroutines, and much more. In addition, **BATCOM** protects your source code, and you can distribute your compiled programs without royalties. Only **$59.95**

ORDER TODAY!

Wenham Software Co.
5 Burley St.
Wenham, Ma. 01984
(508) 774-7036

Circle **119** on Reader Service Card

FREE TRIAL

Pinnacle File Manager

We guarantee immediate productivity gains from using our C file manager library. Simple table-and-column view of data with advanced B-Tree indexing and hi-speed cache frees your brain to *solve application-specific problems.*

Vermont Database Corp.
RR3 Box 6516
Stowe, VT 05672
802-253-4437

Contact this advertiser directly.

FORTRAN

FORTRAN Toolkits
Comprehensive libraries of over 300 graphics, text, and systems service subroutines from ellipses to child-processes to I/O verification. High-speed video drivers, from CGA to Super VGA, are all internal. Option rich, easy to use. No royalties. 300+ pg. manual, 8,000 lines of sample source. Price $215.

DOCUMENT'er and DIAGRAM'er
Makes source code understandable. Indexes, cross-references, and diagrams the structure of source code. No changes to source required. Nested includes, recursion, and annotations handled automatically. Price $165.

IMPULSE *Engineering*
PO Box 190206,
San Francisco, CA 94119-0206
(415) 788-4611

Circle **123** on Reader Service Card

THE CD-ROM DEVELOPER'S LAB™

Comprehensive CD-ROM multimedia production resource for PC and Apple developers. Proven design, management, data prep, programming, premastering and manufacturing techniques. Important specs from leading companies. Demos of off-the-shelf tools for imaging, audio and animation. Sample applications demonstrate the power of our source tools in C, Turbo Pascal.
PC or MAC.@ **$395.00;**
Transportable version @ **$425.00,**
Visa/MC accepted.

SOFTWARE MART, INC.
3933 Spicewood Springs Rd., Suite E-100
Austin, Texas 78759
(512) 346-7887 FAX: (512) 346-1393

Circle **124** on Reader Service Card

"Model the World" With Class

MEIJIN++
C++ Modeling & Simulation Class Library

- **Reusable Data Structures**
- ***Statistical & Math Classes***
- **Configurable Discrete-Event & Scheduling Mechanisms**
- **Handling of Exceptions & Virtual Memory Paging**
- **C++ 2.0 Documented Source**
- **No Royalties. User's Manual**
- **Introductory Price $225.00**

Support for Zortech 2.0+ & Borland TC++ 1.0+

NETWORK INTEGRATED SERVICES, INC.
714-755-0995
221 West Dyer Road
Santa Ana, CA 92707
Fax: (714) 433-2347

Circle **125** on Reader Service Card

dBASE FILE ACCESS FROM C AND C++

Code Base 4.2 is a library of C routines which gives complete *dBASE and Clipper functionality* and file compatibility. The routines correspond directly to dBASE commands. Create multi-user applications using pulldown menus, popup windows and data entry. Code Base 4.2 supports DOS, Unix, OS/2 and Microsoft Windows. Use any C or C++ compiler. $295 with Source! **FREE DEMO.**

Sequiter Software Inc.
P.O. Box 5659, Station L
Edmonton, Alberta
Canada: T6C 4G1
Tel (403) 448-0313 Fax (403) 448-0315

Circle **126** on Reader Service Card

Offer your product directly to Professional Programmers:

Placing an ad in **PJ BILLBOARD** is an easy, low-cost way to reach potential buyers. Display ads are 1⅝" wide by 2⅜" high. For more information or to reserve a space, call our ad department at **(800) 234-0386.**

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

OPT-TECH SORT/MERGE

Extremely fast Sort / Merge / Select utility. Run as an MS-DOS command or CALL as a subroutine.
Supports most languages and filetypes including Btrieve and dBase. Unlimited filesizes, multiple keys and much more!

MS-DOS **$149.**
OS/2, UNIX **$249.**

Opt-Tech Data Processing
P. O. Box 678
Zephyr Cove, NV 89448
(702) 588-3737

Circle 113 on Reader Service Card

QUALITY TOOLS

Memory Overlay Manager - transform your C, Clipper or Clarion application into a 5K TSR with hot-keys.....$99.95
Memory Master - break 640K barrier, one heap or many, EMS/XMS.....$59.95
ASMplus - structured ASM lang. & lib. Code ASM just like C or Pascal....$99.95
"C"erious Toolkit Plus - over 250 highly optimized fcts, C & ASM source...$99.95
386PM protected mode ver.....$199.95
Phase One - 100% replaceable plug-&-play C engine for Paradox, runs 4-10X faster, DOS and DDL ver. incl.....$495.00
PALCom - PAL lang. compiler. Create stand alone .EXE programs.....$1495.00

Please call our BBS at 516-331-6682 to try out our FREE demo programs.

TSR Systems Limited
116 Oakland Avenue
Port Jefferson, NY 11777
Ph: 516-331-6336 FAX: 516-331-6377

Circle 114 on Reader Service Card

BGI PRINTER DRIVERS

Full Source Included!

The BGI Printer Driver Toolkit provides BGI printer drivers for Epson 9 & 24 pin dot matrix, HP LaserJet II, and HP PaintJet printers. Use the BGI graphics interface to effortlessly generate high resolution hardcopy with Turbo C, C++, and Turbo Pascal. Full driver source provided for your tutorial pleasure. $89.95 ppd USA, elsewhere add $4.00. VISA & MC accepted..

RYLE DESIGN
PO Box 22, Mt. Pleasant MI 48804
517-773-0587

Circle 115 on Reader Service Card

The $25 Network

Try the 1st truly low cost LAN

- Connect 2 or 3 PCs, XTs, ATs, PS/2s
- Uses serial ports and 5 wire cable
- Runs at 115K baud, up to 90 feet
- Transfer 8500 bytes per second (ATs)
- Runs in background, totally transparent
- Share any device, any file, any time
- Needs only 14K of ram
- Replace all file transfer software
- $25 per network, NOT PER NODE!
- OVER 15,000 SOLD WORLD WIDE

Skeptical? We make believers!

iModels
Information Modes
Drawer F, Denton, TX 76202
817-387-3339 Technical
1-800-628-7992 Orders

Circle 116 on Reader Service Card

BASIC AND C PROGRAMMERS

INDEX MANAGER ®
B+ Tree File Indexing
Simple, powerful, affordable

New QuickC/TurboC version $99
QuickBASIC version $59
BASIC PDS 7 version $99
Network version $250

CDP Consultants
1700 Circo Del Cielo Drive
El Cajon, CA. 92020
(619) 440-6482

Contact this advertiser directly.

Software Marketing

The *Software Success Reference Book (1987-1988)* is a **MUST READ** if you want to market your software products successfully. Written by David H. Bowen, publisher of *Software Success,*™ the monthly newsletter on successfully running a software business. The *Reference Book* is a 268-page guide organized by topic. Covers Lead Generation, Promotion, Pricing, Distribution, Support, etc. **Only $25.** Check or Credit Card (VISA/MC/AEX).
100% Money Back Guarantee

Software Success
PO Box 9006, San Jose, CA 95157
(408) 446-2504
FAX (408) 255-1098

Circle 120 on Reader Service Card

DISK DISTRIBUTOR

SOFTWARE DISTRIBUTORS
SITE LICENSEES
USERS GROUPS

AUTOMATICALLY DUPLICATE AND LABEL DISKETTES WITH YOUR PC. NO EXTRA HARDWARE REQUIRED.

Use your PC to duplicate software distributions of any quantity of disks AND print labels. Duplicates any combination of 3 1/2" & 5 1/4" disk formats. Stores disk images on hard disk for unlimited copying. NCP.

INTRODUCTORY PRICE $79

ORION SOFTWARE CO.
P.O. Box 1913
Bellingham, WA 98227 U.S.A.
Tel: 206 647-2138 Fax: 206 734-7328

Circle 121 on Reader Service Card

UPPER DECK FORTH

$49

Based on the Forth-83 Standard. Fully segmented architecture. Fast compile, no link required. Uses normal DOS files. Includes editor, assembler, decompiler and debugger. Turnkey application support. For PC's and compatibles, 256K, DOS 2.0 or later. Add $3 for shipping and handling (outside USA $15). CA residents add sales tax.

UPPER DECK SYSTEMS
P.O. Box 263342
Escondido, CA 92026
(619) 741-1075

Circle 122 on Reader Service Card

Real-time programming is tough... unless you have SynRTX

SynRTX is the most comprehensive object-based Real-Time eXecutive library (where each object is a task) available for C and C++. It allows your PC to perform hundreds of tasks simultaneously at a simplicity you just wouldn't believe. A complete reference manual and application examples show you how to make it fly.

No charge for AIR shipping anywhere in the world. 30-day money-back guarantee. 1-year upgrade. No royalties, and free technical support.

Low price, simplicity, speed, and service. It's a combination you can't afford to miss. So act today -- at $79, there's no reason to wait!

If you call our fax NOW, our free brochure that explains all the confusion with today's real-time buzz words will be mailed to you within 24 hours.

Get also our real-time malloc package: the fastest memory allocation/deallocation based on object's lifetime. Replace your malloc, and speed up your program, for just $29.

Cnapse The Real-Time Contact
(514)447-7221 Fax:(514) 447-7297
11 Laurier, Suite 103, Chambly, Qc, Canada J3L 9Z7

Circle 127 on Reader Service Card

MAKE $$$ FREE

Device Driver Inc. would like to market device driver programs that have been developed by your company. These programs have significant revenue potential.

Drivers are marketed through our catalog, which is distributed at trade shows & through direct mail to systems integrators, manufactures of computer peripherals, VARS, consultants and other interested parties. If you have written drivers for PC's, workstations, midrange or super computers and are interested in having them listed in our catalog, for free, call **(401) 724-8756** or write to:

DEVICE DRIVERS, INC.
990 Mineral Springs Ave.
No. Prov., RI 02904
Fax # (401) 726-8940

Circle 128 on Reader Service Card

A Software Development Essential!

DOCZ™

Save up to 35% of your development time with DOCZ to automate subroutine and program documentation. Take advantage of the object-oriented technology of DOCZ to share and re-use your software modules. FREE MSDOS demo disk and samples. 30-day money-back guarantee. VISA/MC accepted. $195/Single-User. Start benefiting NOW!

Do It Better With Toolz!
Software Toolz, Inc.
8030 Pooles Mill Drive
Ball Ground, GA 30107-9610
(404) 889-8264
Orders, V-mail: (800) 869-3878

Circle 129 on Reader Service Card

INTEL® 80386 MACRO DISASSEMBLER

MD86 (Masterful Disassembler) is the most comprehensive macro disassembler on the market. Interactive by design not after-thought. Features include:

- ► User defined macros with arguments
- ► Auto code/data separation
- ► Auto commenting and label naming
- ► Command menus and help screens
- ► Customizable disassembly options
- ► Comprehensive, indexed, manuals
- ► Fully supports 8086/87 thru 80386/87
- ► COM/EXE/SYS/memory/other
- ► Cross reference and more!

Still only $67.50 + tax ($1.50 s/h)

C.C. SOFTWARE
1907 ALVARADO AVE.
WALNUT CREEK, CA 94596
(415)939-8153

Circle 130 on Reader Service Card

Coming Next Issue: HARDWARE FOR DEVELOPERS

May/June 1991 Vol. 9.3
Advertising space reservations are due:
March 1, 1991
Materials are due:
March 8, 1991

Please call 800-234-0386

Ask for Jay or Robin in the advertising department to reserve your space.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Changes in the fast lane

By Hal W. Hardenbergh

Heraclitus asserted 2400 years ago that "nothing endures but change." He must have known about what's coming this year with 486 PC clones.

December 10 marked the first time a mass marketer pushed an under-$2000 486 machine in a mainstream PC publication. That price includes everything but a hard disk and display system. By the time you read this, that price may seem high.

You can find 486 machines for $1500 and up, with no apparent limit to "up." I don't remember such a vast spread of prices at this point on the 386's development curve, so that's one change in the market. But there's another change, a big one, that will be evident by the end of '91. You're going to be seeing 44-MHz 486 machines sold for the same price as the lowest-cost 25-MHz machines. Honest. Here's why.

In my job as a hardware designer, I use a lot of programmable logic devices (PLDs) from a certain manufacturer. When I first started using them, they ran at 25 nsec propagation delay. Later they improved to 15 nsec, and last year to 10 nsec. I expect 7-nsec parts from this company soon. They still sell the 25- and 15-nsec parts, at very favorable prices.

JUST A NSEC, NOW

How, you may ask, does this company split production between 10- (or 7-) nsec parts and 25-nsec parts? It doesn't. All parts are made on the same production line using the same process.

Let's say that the current production orders are for 100,000 10-nsec parts, 75,000 15-nsec, and 40,000 25-nsec ones. The company tests raw dies until it has 100,000 10-nsec parts. The yield is very high, and most parts that won't run at 10 nsec won't run, period. They then take the remaining untested dies and the few functional parts that failed the 10-nsec test and run their tester at 15 nsec until they have filled their orders for 75,000 15-nsec parts. Finally, they test for 40,000 25-nsec parts.

More than 90% of the 15- and 25-nsec parts are, in fact, fully functional 10-nsec parts. They simply have not been

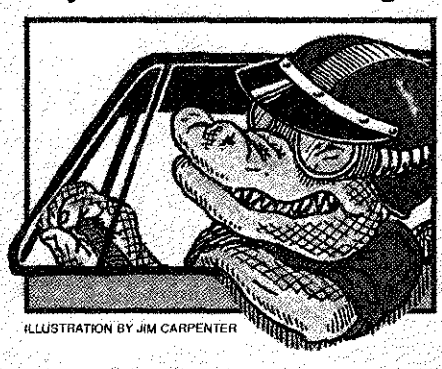


ILLUSTRATION BY JIM CARPENTER

tested and certified at that speed. Why sell 10-nsec parts for 25-nsec prices? What's the company to do? Maintain a second production line just to build slow parts? Turn down the orders for 25-nsec parts from folks who designed them into their equipment three years back? No, this is how chip manufacturers do business.

FAST, FASTER, FASTEST

Microprocessors are also manufactured, tested, and marketed in exactly this manner. I just got my hands on some recent date-code 68030s marked "20 MHz." Wow, them parts am fast! In fact, just about any recently manufactured 68030 will easily run at 50 MHz.

The speed of a CPU depends on the design rules used (1.5 to 0.8 micron) and the maturity of the manufacturing process. There is a big jump in speed when the die is shrunk and then a gradual increase as the manufacturing process stabilizes, mainly because the 16 successive mask layers required of a CMOS process are better aligned as time passes. *The actual speed variation of currently manufactured CPU dies is unrelated to the speeds marked on the chip package.*

The 386 is built using a 33-MHz process and has been for a long time. That process is mature. Almost any current-production 386 will run at 33 MHz (or faster). Unplug a current date-code 16-MHz 386 and plug it into a 33-MHz motherboard. Watch it run perfectly.

Clone makers have not been taking advantage of this speed because to run a 386 at 33 MHz, a 33-MHz memory system is needed. The cost of a 33-MHz 386 system is mostly in the memory, not in the CPU. Once you've spent the money on the memory, it doesn't make sense to try to save a few bucks on the CPU.

Intel will start shipping 50-MHz 486s in 1991. By the end of the year, virtually every 486 being shipped will in fact be a 50+ MHz CPU, no matter what speed is stamped on the package. It will become perfectly reasonable and rational for the clone-makers to run "25-MHz" 486 chips at 50 MHz. I happen to think it will also become common.

But what about the 50-MHz memory? Ah. The 486, running at 50 MHz, has its own 50-MHz no-wait-state memory. The 8K internal cache, remember? The clone-makers will take a 25-MHz 486 machine with normal, inexpensive 25-MHz DRAM memory. They will provide a "turbo switch" to optionally enable a 50-MHz clock for the CPU. When there is a cache miss—6% of the time?—the cache line is filled from 25-MHz memory at 25-MHz speeds. When there is a cache hit—94% of the time?—the CPU runs at 50 MHz. The average will be around 44 MHz at no extra cost, except for the turbo switch.

MAKE MINE TURBO

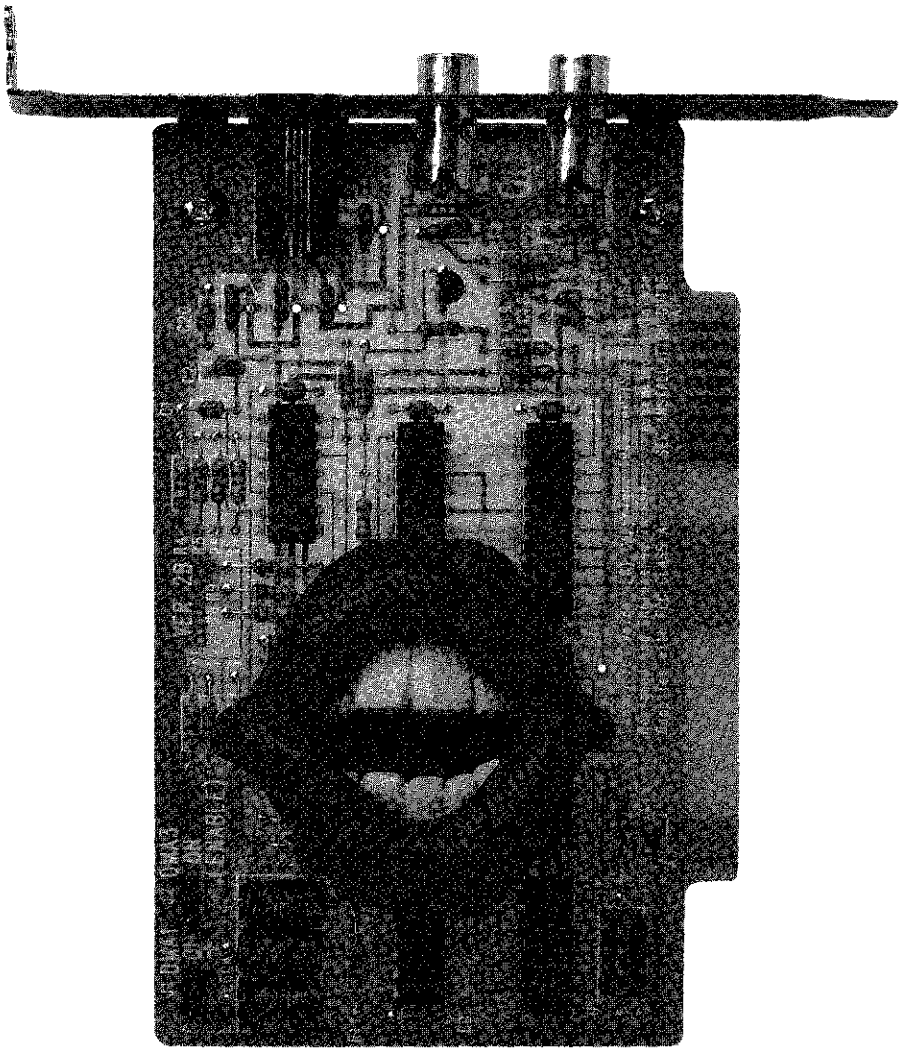
This arrangement will probably be more reliable than previous "turbo" boosts because there is no change in the timing to the memory system or the system bus. In past years, some turbo machines malfunctioned because the disk controller on the system bus couldn't run at the faster clock rate.

Would you rather pay clone prices for a machine that runs at 44 MHz or mortgage your house to buy a genuine 50-MHz machine from a brand-name supplier? Those will be your choices at the end of this year, unless you really want to keep your 6-MHz AT.

Some organizations definitely aren't going to like this soon-to-transpire state of affairs. Heraclitus had something to say about this, too: "From things that differ comes the fairest attunement; all things are born through strife." Sounds like a free-marketeer to me. PJ

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Make your software laugh, sob, growl, warble, wail, sing and talk.



If you've ever wanted to add sound to your software, put on your headphones. The LANtastic™ Voice Programmers Interface (VPI) and the LANtastic Voice Adapter are waiting.

Now you can create educational and training programs on disk or CD-ROM that actually speak to students.

Or voice mail packages that send speech across a LAN.

Or "talking" software demos.

Music synthesis.

Sound effects.

And whatever else echoes through your imagination.

You see, the LANtastic Voice Adapter is a low-cost expansion device that converts between audio signals and a digital data stream. It has mono tape in and out jacks so you can record from a CD player, tape recorder, microphone or other external sound source, then play back through an amplifier with speakers, or through the telephone handset provided with each adapter.

Better yet, the LANtastic VPI is a set of simple, straight-forward software tools, which make it possible for your applications programs to use the LANtastic Voice card to record and play digitized sound. And no one else offers anything like it. Especially at $149* per adapter and $195 for the VPI.

No wonder *PC Magazine* (July 1990) wrote: "Leonardo DaVinci might have called it a work of art." *And:* "LANtastic's strong VPI makes it a natural choice for developers wishing to add another dimension to their communications."

For complete information about Artisoft's Voice Developers Kit, call 602-293-6363.

ARTISOFT

*Manufacturer's suggested retail price. ©1990 ARTISOFT. LANtastic is a trademark of Artisoft. Brand names and product names are trademarks or registered trademarks of their respective companies.

Circle 55 on Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Don't miss the Borland Languages Conference.
April 28 through May 1, 1991, San Francisco, CA. For information, call 1-800-942-TURBO

Borland's Turbo Pascal 6.0 is the Fastest Way from Inspiration to Application

Jump-Start Your Application

When you're inspired to write a program, you want to spend your time developing code that solves your problems. Not hours and hours writing common routines for event handling, data management or user interface.

Now, you can jump-start your applications development by programming with the latest release of the World's #1 Pascal Compiler, Turbo Pascal® 6.0 with Turbo Vision.™

Now with Turbo Vision

With Turbo Vision, the first object-oriented application framework for DOS, you get a giant head start on creating better applications in far less time. Use a Turbo Vision object and your program automatically inherits a hot program architecture that includes overlapping windows, pull-down menus and mouse support. Turbo Vision makes it fast and easy—setting you free to develop the parts of your applications that solve your problem.

And Turbo Pascal 6.0 comes loaded with Turbo Vision applications including a calendar, a calculator, an editor, a clock, a directory browser and forms.

New Turbo-Charged Environment

The new Turbo Integrated Development Environment (IDE) features a multi-file editor, overlapping windows and mouse support.

And Turbo Help lets you copy, compile and run an example program for every standard Pascal library routine so you can use it in your code.

Pro Version with Turbo Drive™

Turbo Pascal® Professional 6.0 also includes: a professional version of the compiler with Turbo Drive,† for compiling big applications in extended memory; Turbo Debugger® 2.0, for killing the toughest bugs; Turbo Profiler™ 1.0, for eliminating bottlenecks; and Turbo Assembler® 2.0, the world's fastest, 100% MASM-compatible assembler.

An Inspiring Offer

REGISTERED TURBO PASCAL OWNERS: You can upgrade to Turbo Pascal 6.0 for only $69.95* or to Turbo Pascal® Professional 6.0 for only $99.95* direct from Borland. The suggested retail price for Turbo Pascal 6.0 is $149.95, and $299.95 for Turbo Pascal Professional 6.0.

To order, see your dealer
To upgrade, call now: 1-800-331-0877

BORLAND

The Leader in Object-Oriented Programming

Circle **56** on Reader Service Card

CODE: MF04

*Add $10.00 for shipping and handling. Offer expires August 31, 1991. Offer good in U.S. and Canada only. Mail orders to: Borland International, Inc., P.O. Box 660001, Scotts Valley, CA 95067-0001. Residents in CA, CT, GA, IL, MA, MI, NY, OH, PA, TX, VA and WA, please add appropriate sales tax. For orders outside the U.S., call (408) 438-5300. †Turbo Drive compiler requires 1Mb extended memory. Hard disk recommended. Copyright ©1990 Borland. All Borland products are trademarks of Borland International, Inc. Corporate Headquarters: 1800 Green Hills Road, P.O. Box 660001, Scotts Valley, CA 95067-0001, (408) 438-5300. Offices in Australia, Denmark, France, Germany, Italy, Japan, Singapore, Sweden and the United Kingdom. Porsche is a registered trademark of Porsche. BI 1391B

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED